YOUR COMMODORE £1.50

SERIOUS USERS GUIDE 1987

THE ULTIMATE GUIDE FOR COMMODORE OWNERS

INCLUDING:

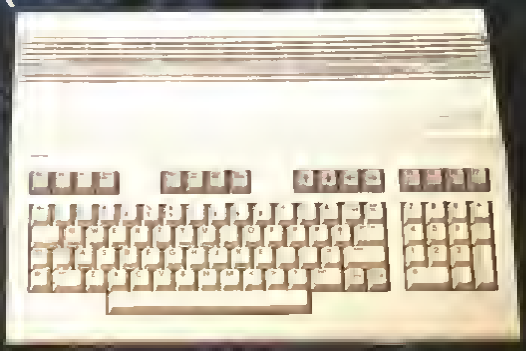
A SUPERB C64 80 COLUMN WORDPROCESSOR

PLUS/4 EXTENDED BASIC

PROTECTING YOUR PROGRAMS FROM PRYING EYES

NEW CHARACTER SETS FOR YOUR MPS801/3

TECHNICAL INFORMATION FOR THE C64, C128, PLUS/4 AND C16



FROM THE PUBLISHERS OF YOUR COMMODORE

THE PARALLEL DISK TURBO SYSTEM EVERY 1541 WANTS!

THE NEW! PHANTOM

NOW WITH ADDED POWER!

- LOADS 249 BLOCKS IN 7secs
- LOADS 64 BLOCKS IN 2secs
- SAVES 64 BLOCKS IN 4secs

across bus with FULL ERROR CHECKING

- FORMAT 35 TRACKS IN 21secs
- FORMAT 40 TRACKS IN 23secs

with TRACK by TRACK VERIFY

FREE DISK COPIER WORTH £20.00 WITH EVERY PHANTOM

This copier uses the Phantom hardware to recreate bit by bit a duplicate of whatever unprotected disks.

- EXPERT COMPATIBLE – PROGRAM THE EXPERT IN LESS THAN 2secs.
- SPEEDS UP ALL DRIVE FUNCTIONS
- FULL ERROR CHECKING RETRIDEO – ESSENTIAL FOR RELIABILITY

Other systems sacrifice this CRUCIAL FUNCTION to achieve speed increases but are in fact NO FASTER only LESS RELIABLE Buy them systems and you'll find out the hard way!

- 8 FUNCTION KEY COMMANDS FOR MAJOR FUNCTIONS.
- 60 ADDITIONAL COMMANDS INCLUDING
- FILE LOCK & UNLOCK
- WRITE PROTECT IGNORE – NO NEED TO NOTCH DISKS
- SET DEVICE NUMBER ● SCREEN ON/OFF
- CD15&T – REVERTS TO STANDARD 1541
- MANY OTHER USEFUL COMMANDS AND COMPATIBILITY (8 extra function keys, disable extra rom etc.)
- DRIVE MONITOR – COMMANDS INCLUDE DISASSEMBLE FILL COMPARE HUNT ASSEMBLE EXECUTE ETC. ETC.
- COMPREHENSIVE 40 PAGE MANUAL & ILLUSTRATED FITTING GUIDE
- COMPATIBLE WITH MOST COMMERCIAL SOFTWARE
- RELIABLE – DOES NOT CORRUPT DISKS UNLIKE RIVAL PRODUCTS
- SWITCHABLE KERNAL REPLACEMENT INCLUDED FOR 64 OR 128 GAMEDEV
- BRITISH DESIGNED AND MANUFACTURED – AVAILABLE EXCLUSIVELY FROM TRILOGIC
- UPGRADABLE – OS KERNAL & COPIER UPGRADES WILL BE AVAILABLE SOON TO GIVE EXTRA FEATURES FOR A NOMINAL SUM

OPTIONAL EXTRAS

DRIVE COOLING KIT 1 PRICE £2.50

[illegible]

DRIVE COOLING KIT 2 PRICE £17.95

Consist of a very quiet mains operated 80mm fan and [illegible] to the drive. No soldering [illegible]

PRICE £78.99 inc VAT & p&p

THE EXPERT CARTRIDGE WITH BUILT IN E.S.M.!

- SAVES PROGRAMS IN ONE FILE – (incl. multipart ones)
- COMPACTS PROGRAMS used by leading software houses.
- RELOADS DECOMPACTS & RUNS ANY PROGRAM WITHIN 35 secs.
- THE EXPERT IS NOT NEEDED FOR RELOADING
- COMPATIBLE WITH THE PHANTOM II ALL CBM64/128 DISK DRIVES.

WITH THE NEW V2.10 SOFTWARE SUPPLIED, THE EXPERT CAN DO ALL THIS AND MORE

PROGRAMME PARALYSER Stops even the most heavily protected programs [illegible] data with [illegible] techniques

BACK UP GENERATOR TAPE/DISK ● DISK TAPE ● DISK DISK ● TAPE TAPE (Transfers memory resident) [illegible]

SPRITE EXTRACTOR & IMMORTALISER

HIRES SCREEN GRABBER

CHEAT MACHINE

CODE INTERROGATOR

THE ONLY PROGRAMMABLE CARTRIDGE

TRILOGIC

PRICE £29.99 inc VAT & p&p

EXPRESS DELIVERY ADD £1.96

FAST MAIL ORDER SERVICE ● PROMPT DESPATCH ● ALL PRICES FULLY INCLUSIVE PLUS 10 DAY MONEY BACK GUARANTEE ON ALL BACK UP DEVICES ORDERING: WRITE OR PHONE PAYMENT BY CASH, CHEQUES PAYABLE TO TRILOGIC, BY POSTAL ORDER OR ACCESS ● ADD £1 EXTRA FOR EXPORT ORDERS PAYMENT IN STERLING ONLY PLEASE

DEPT PX 3 329 TONG STREET BRADFORD BD4 9QY TEL (0274) 691115

YOUR COMMODORE SERIOUS USERS GUIDE 1987

Editor Stuart Cooke
Assistant Editor Sue Joyce
Editorial Assistant Kirk Rutter
Senior Advertising Manager Pete Chandler
Advertisement Manager Stuart Taylor
Advertisement Copy Control Laura Champion
Typesetting Project 3
Design ASP Art Studio

Argus Specialist Publications Ltd., Editorial & Advertisement Office, Your Commodore, No. 1 Golden Square, London W1R 3AB. Telephone 01 437 0626. Telex 8811896

CONTENTS

The Your Commodore Serious User Guide is packed full of vital information and programmes for all types of Commodore owners

If you use your computer for 'home office' purposes then the 80 column C64 wordprocessor will no doubt come in extremely handy (tape and disk supported) If you need to keep lists of information, the database Multifile will be very useful Multifile is customized to suit your specific needs and it can be used for anything from running a stock control to keeping a list of names and addresses

Many utility programmes are also included MPS801/3 owners can now use descenders and user defined character sets on their printers. Plus 4 owners can add a wealth of new commands with our extended Basic Disk owners will find the fast formatter and file describer invaluable utilities

If you write your own programmes then there's plenty in the Guide for you too Program Lock will protect your programmes and keep prying eyes from reading your code A character scroller for the C64 will help to improve the visual effect of your programs

Beginners and hardened programmers alike will find the wealth of technical information provided in our Technical Appendix, an invaluable reference Here you will find memory maps for all of the popular Commodore computers ROM calls are also listed so that you can find out, at a glance, information that you need when programming Aids such as the hex decimal convertor and the list of useful POKE commands will also prove extremely useful

The Your Commodore Serious Users Guide is something that no Commodore owner should be without.

Listings

Get it right first time with our deluxe program system for the C64.

You may have noticed that our listings are free of those horrible little black blobs which send you searching around the keyboard for a suitable graphic symbol. You may also have noticed the funny numbers by the side of each line of the listing. Fret no more – it's all part of our easy entry aid.

Instead of those nasty graphics and rows of countless spaces in PRINT statements and strings we use a special coding system. The code, or mnemonic, is always contained in square brackets and you'll soon learn to decipher their meanings.

For example [SA] would mean type in a Shifted A, or an ace of spades in layman's terms, and [SA10] would mean a row of ten of these symbols.

[S+2] means hold down the shift key and press the plus key twice. It doesn't take a great leap of logic to realise that [C+2] means exactly the same thing except that the Commodore key (bottom left of the keyboard) is held down instead of the shift key.

If more than two spaces appear in a statement then this will be printed as [SPC4] or, exceptionally, [SSPC4]. Translated into English this means press the spacebar four times or in the latter case hold the shift key down while you do it.

A string of special characters could appear as:

[CTRL N, DOWN2 LEFT5,BLUE F3 C3]

This would be achieved by holding down the CTRL key as you press N, press the cursor key down twice, the cursor left key five times, press the key marked BLUE while holding down the CTRL key, press the F3 key and, finally, hold the Commodore key down while pressing the number two key (C2 would of course make the computer print in brown).

Always remember that you should only have a row of graphics characters on your screen with no square brackets and no commas, unless something like this appears:

[SS][C*]

In this case the two characters should have a comma between them.

On rare occasions [REV T] will appear in a listing. This is a delete symbol and is created by entering the line up to this mnemonic. Then type a closing quotation mark (SHIFT & 2) and delete it. This gets the computer out of quotes mode. Hold down CTRL and press the number nine key (RVSON), type the relevant number of reversed T's and then hold down CTRL and press zero (RVSOFF). Next type another quotation mark and delete it again. Now finish the line and press RETURN.

A list of these special cases is given in the table but remember that only one of these mnemonics will appear outside of a PRINT string, the symbol for pi. This may appear when its value is needed in a calculation so this may look something like:

CC=2*[PI]*R

Ignore the square brackets and just type in a shifted upward pointing arrow (ie the pi symbol).

```
PROGRAM: SYNTAX CHECKER

 6 REM SYTAX CHECKER - ERIC DOYLE

 10 BL=10    LN=70    BA=49152
 20 FOR L=0 TO BL CK=0 FOR D=0 TO
    15
 30 READ A IF A>255THENPRINT NUMB
    ER TO LARGE" LN=LN+10 STOP
 40 CK=CK+A POKE BA+L*16+D A:NEXT
    D
 50 READ A IF A<>CK THENPRINT"ERR
    OR IN LINE";LN+(L*10) STOP
 60 NEXT L SYS 49152 NEW
 70 DATA 173,5,3,201,100,208,31,1
    69,9,141,32,208,141,33,208,1
    6947
 80 DATA 169,7,141,134,2,169,13,3
    2,210,255,169,64,141,4,3,169,160
    2
 90 DATA 192,141,5,3,88,96,120,16
    9,124,141,4,3,169,165,141,5,1588
100 DATA 3,169,14,141,134,2,141,
    32,208,169,0,141,33,208,96,15
    85
110 DATA 32,164,185,72,138,72,15
    2,72,162,0,165,20,133,254,165,01
    ,1747
120 DATA 24,101,254,133,254,160,
    0,2,240,18,69,254,133,254,232,18
    9,2346
130 DATA 0,2,240,8,24,101,254,13
    3,254,232,208,233,169,1,141,134
    ,2134
140 DATA 2,165,254,74,74,74,74,3
    2,150,192,32,210,255,165,254,41
    ,2054
150 DATA 15,32,150,152,32,210,25
    5,169,13,32,210,255,169,13,32,01
    ,1595
160 DATA 255,169,7,141,134,2,104
    ,168,104,170,104,96,24,105,48,20
    1,1432
170 DATA 58,16,3,56,24,105,7,96
    ,0,0,0,0,0,0,0,403

by Eric Doyle
```

Mnemonic	Symbol	Keypress
[RIGHT]		CRSR left/right
[LEFT]		SHIFT & CRSR left/right
[DOWN]		CRSR up/down
[UP]		SHIFT & CRSR up/down
[F1]		f1 key
[F2]		SHIFT & f1 key
[F3]		f3 key
[F4]		SHIFT & f3 key
[F5]		f5 key
[F6]		SHIFT & f5 key
[F7]		f7 key
[F8]		SHIFT & f7 key
[HOME]		CLR/HOME
[CLR]		SHIFT & CLR/HOME
[RVSON[		CTRL & 9
[RVSOFF]		CTRL & 0

Mnemonic	Symbol	Keypress
[BLACK]		CTRL & 1
[WHITE]		CTRL & 2
[RED]		CTRL & 3
[CYAN]		CTRL & 4
[PURPLE]		CTRL & 5
[GREEN]		CTRL & 6
[BLUE]		CTRL & 7
[YELLOW]		CTRL & 8
[POUND]		£
[LARROW]		←
[UPARROW]		↑
[PI]		SHIFT & ↑
[INST]		SHIFT & INST/DEL
[REV T]		see text
[Cletter]		CBM + letter
[Sletter]		SHIFT + letter

Checksum Program

The hexadecimal numbers appearing in a column to the left of the listing should not be typed in with the program. These are merely checksum values and are there to help you get each line right. Don't worry if you don't understand the hexadecimal system, as long as you can compare two characters on the screen with the corresponding two characters in the magazine you can use our line checking program.

Type in the Checksum Program, make sure that you've not made any mistakes and save it to tape or disk immediately because it will be used with most of the present and future listings appearing in Your Commodore.

At the start of each programming session, load Checksum and run it. The screen will turn brown with yellow characters and each time you type in a line and press the RETURN key a number will appear on the screen in white. This should be the same as the corresponding value in the magazine.

If the two values don't relate to one another, you have not copied the line exactly as printed so go back and check each character carefully. When you find the error simply correct it and press RETURN again.

If you want to turn off the checker simply type SYS49152 and the screen will return to the familiar blue colours. You can then do whatever it was you wanted to do and if this doesn't use the area where Checksum lies you can go back to it with the same SYS command.

No system is foolproof but the chances of two errors cancelling one another out are so remote that we believe our listings are more reliable than any other magazine in the world. So get typing!

Hashing it with Commodore

We would all like to make more use of data files but lack of clear advice as to how to index/retrieve the data often deters programmers from using relative files to their full advantage

By Owen C English

A problem often encountered with databases employing relative files is that of not being able to rapidly read records without specifying the DOS record number. Clearly searching and comparing in record number sequence through an entire data file in order to find one record defeats the purpose of random access files. Imagine being able to access a record by defining the data of one or a combination of more than one field.

For example if you have a relative data file containing six fields

SURNAME
FIRSTNAME
BIRTHDATE
STREET
TOWN
COUNTY

You may need to find a record by specifying the SURNAME and FIRSTNAME without even knowing by what record number the data had been filed. The purpose of a good database would preclude you keeping a list of record numbers and records so it would be most unlikely that you would know the DOS record number anyway.

The solution to this problem of how to find records by specifying the data lies in the maintenance of one or more *HASH FILES*. Yes, you may say "why HASH a program that's probably already a HASH? well hashing doesn't quite mean to make a mess of it!

One creates a number called a *hash number* by performing a mathematical calculation upon the data in defined data fields. That *hash number* (referred to hereafter as *HASH(No.)*) is ideally unique to any set of data. Let us take for example, the following record and perform a calculation upon it.

The record could be as shown in Figure 1.

A short Basic program, normally a subroutine, such as that in Figure 2 could be used to work out the *HASH(No.)*.

NOTE although I have used variable names of more than two characters length Commodore Basic will only recognise the first two letters.

The variable MF (multiplication factor) is calculated to give the appropriate range of *HASH(No.)* numbers and the SF (subtraction factor) lowers the range minimum to zero. The range must generally be twice the total number of possible records to be written – this reduces the chance of a double up of the unique *HASH(No.)* numbers.

Lets us take another working example. If we apply the formula in the program in Figure 2 we will find that the name HENRY BLOGGS produces a *HASH(No.)* of 791. The name JULIA FORSE produces a *HASH(No.)* of 1359.

Writing data

Right how do we use this *HASH(No.)*? We find the next available record number (from a sequential file named LASTUSED.DAT) and write the six fields of data into that record number in the main datafile, MAINDATA.DAT. We then calculate *HASH(No.)* and write the used record number as DATA into record no. *HASH(No.)* in the index file INDEX.DAT.

Well, that may seem complicated but by the careful use of files a very retrievable database can be structured.

Getting it back

Reading records is a reverse of the above although a little simpler the user is asked for the SURNAME$ and FIRNAM$ of the record that is required to be retrieved. The *HASH(No.)* is then calculated with exactly the same formula. It will be the same as it was when the record was written as nothing has changed in the calculations. The data is then read from record number *HASH(No.)* in INDEX.DAT. That data will be the record number that the six fields of DATA were written to in MAINDATA.DAT so it only remains to read that data from MAINDATA.DAT.

If you can foresee that you may need to search for data by other fields or combinations of fields, more index files

Figure 1

FIELDNO	FIELDNAME	FIELDTYPE	DATA
1	SURNAME$	ALPHA	BLOGGS
2	FIRNAM$	ALPHA	HENRY
3	BIRDAT$	ALPHA	200953
4	STREET$	ALPHA	56 THE CLOSE
5	TOWN$	ALPHA	HORNCHURCH
6	COUNTY$	ALPHA	ESSEX

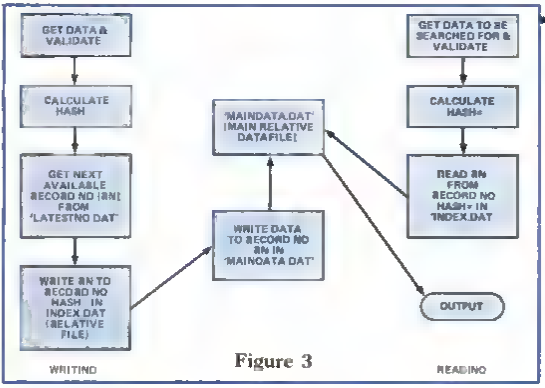


Figure 3

could be maintained – these could even be created at a later date by reading the MAINDATA DAT file sequentially and creating additional index hash files

Duplicates

Now, one obvious problem is that of a *HASH(No)* occurring more than once To safeguard against this possibility, before writing to record number *HASH(No)* in INDEX DAT, read that record to ensure that no data exists If it does , go down and read the next record and when you find a blank one use that Accordingly this means that after looking up a record in INDEX DAT and reading the pointed record from MAINDATA DAT, you must compare it with the specified data to ensure that it is the correct one If it is not go back to INDEX.DAT and read the next *HASH(No.)* down and read the record pointed to in MAINDATA DAT Keep doing this until you find the match

Mathematics will show you that the chances of a double up of a *HASH(No.)* are unlikely as long as you have a maximum *HASH(No)* of two times the maximum number of records to be written into MAINDATA DAT

Getting MF and SF

To calculate MF and SF one must define the characters that we are going to hash and index against Take the example given The maximum *HASH(No)* will occur if the name is ZZZZZZ ZZZZZZ (SURNAME$ and FIRNAM$ respectively) The minimum *HASH(No.)* will occur if the name is AAAAAA AAAAAA (SURNAM$ and FIRNAM$ respectively)

To calculate the range of possible *HASH(No)*, one must decide the maximum number of records to be written to MAINDATA DAT Lets take the maximum here as 1200 records We therefore require a *HASH(No.)* range of 0000–2400 and as each record in INDEX DAT will occupy four bytes (2400 has four characters) the total space to be occupied by INDEX DAT will be 2400*4=9600 bytes

The variable SF is chosen to move the range of minimum *HASH(No)* – maximum *HASH(No.)* to 0000–2400

O.K this may all seem rather complex and time consuming but if you have any doubts to the speed of this system try the following

Set up a dummy data file with six fields of random dummy data in each record(have 1000 records) Hide a record of known data near the end of it at say record number 941 Now, OPEN and READ the file in sequential order from record number one and at each record compare the read data with that known to be "hideing

Go away and have lunch and if you re lucky the experiment may have found the "hidden" record by the time you return

Try setting up a small system as described above with an indexed HASH file and RUN it You will now appreciate what relative data files and good indexing is all about!

Multiple hash files

A word of caution Before deciding to index every field or every combination of fields consider what the most likely unique fields will be Index files occupy a significant amount of disk space and I found it unnecessary In the example database to index more than two combinations I have indexed SURNAM$/FIRNAM$ together and SURNAM$/BIRDAT$ together as it is very unlikely to find two BLOGG s with the same BIRDAT$ In fact on a base of 22000 records, no two have ever matched in that way So despite Commodore s deploringly slow DOS quite a workable database has been created

The simple flowchart in Figure 3 summarises the use of hash files, both writing or creating and reading

Figure 2

```
10   SURNAM$="BLOGGS"  FIRNAM$="HENRY"
20   HASHSTRING$=LEFT$(SURNAM$,3)+LEFT$(FIRNAM$,3)
30   PRINT HASHSTRING$  REM IT SHOULD BE "BLOHEN"
40   FOR I=1TO6 H(I)=ASC(MID$(HASHSTRING$,I I+1)),NEXT I
50   MF=6:64|E-11  SF=724  REM SEE TEXT
60   HASH(No )=(INT((H(1)*H(2)*H(3))+(H(4)*H(5)*H(6))*MF))-SF
     REM DO THE CALCULATION
70   PRINT HASH(No )  REM IT SHOULD BE 791
```

VARIABLES:

H(1) Is the left most character form the string SURNAM$
H(2) Is the second character from the string SURNAM$
H(3) Is the third character from the string SURNAM$
H(4) Is the left most character from the sting FIRNAM$
H(5) Is the second character from the string FIRNAM$
H(6) Is the third character from the string FIRNAM$

Fast Formatter

Fed up with waiting for your disks to be formatted? Speed things up with this handy program

By Ben Wellbay

Fast Formatter is a utility that quite simply, fully formats a disk with ID in a fraction under 10 seconds.

The routine will no doubt gain most benefit when included within a database/filing program that calls upon a disk routine before saving file data. Provision for listing the directory of a disk without destroying a program in memory is also made.

Getting it in

The program is presented here in the form of a Basic loader. This should be typed in using the *SYNTAX CHECKER* program that can be found on the *LISTINGS* page of this magazine. Once you RUN the program it will POKE the necessary machine code into memory. Should you want to SAVE the machine code for later retrieval by another program etc. then you can do so with the following instructions.

POKE43,0 POKE44,202 POKE45,168 POKE46,207:SAVE"ML",8

Using the program

The routine is screen-option driven and should cause no problems. To activate the routine a SYS call is required. To start the program simply type

SYS 51712

Now you can format a blank disk without the normal 80-second wait!!

```
PROGRAM: FAST FORMATTER

30  4 REM * 10 SECOND FORMATTER *
25  0 REM * RESIDENT IN HIGH MEM *
6F  12 REM * TO ACTIVATE ROUTINE *
F2  10 REM *       SYS 51712    *
EE  20 REM *  WILL RESET TO WARM *
C3  24 REM * START WHEN FINISHED *
0E  28 REM *  THUS KEEPING BASIC *
93  32 REM *    MEMORY INTACT   *
C9  36 REM * BY BEN WELLBAY 1988 *
DA  40 REM * FOR "YOUR COMMODORE" *
81  44 FOR I=51712TO53160 READ P:POKE I,A:C=C+A:NEXT
F1  48 IFC<>176810THENPRINT"ERROR IN DATA!":END

10  50 DATA 76,126,205,105,100,201,0,170,4,169,0,133,100,169,5,102
73  51 DATA 207,160,202,32,134,202,32,169,0,162,183,160,203,32,134,202,169
E2  52 DATA 7,162,185,160,204,32,134,202,169,1,162,152,160,0,141,202
79  53 DATA 202,142,200,202,140,201,202,169,07,141,199,202,32,123,202,160
93  54 DATA 0,185,197,202,32,168,255,200,192,8,208,245,160,0,173,167
78  55 DATA 2,32,168,255,32,174,255,162,0,160,5,142,192,202,140,193
00  56 DATA 202,32,123,202,160,0,185,189,202,32,168,255,200,192,5,208
90  57 DATA 245,32,174,255,234,32,123,202,78,174,259,165,160,32,177,255
93  58 DATA 169,111,32,147,255,96,141,185,202,134,251,132,252,160,0,140
D4  59 DATA 164,202,32,123,202,162,0,189,181,202,32,168,255,232,224,6
```

```
AF  60 DATA 208,245,162,32,177,2
    51,32,160,255,200,202,208,24
    7,32,174,255
E3  61 DATA 192,0,208,219,96,77,
    45,87,224,7,32,234,234,77,45
    ,69
F3  62 DATA 0,5,234,234,234,77,4
    5,87,132,0,1,234,234,234,234
    ,173
88  63 DATA 121,5,133,10,173,122
    ,5,133,19,169,1,133,34,169,1
    0,133
A5  64 DATA 106,120,173,0,20,9,4
    ,141,0,20,169,45,133,74,32,1
    47
00  65 DATA 5,190,74,200,249,162
    ,0,32,189,5,32,189,5,169,230
    ,141
E2  66 DATA 12,20,32,0,6,133,192
    ,173,0,20,41,251,141,0,20,18
    9
8Z  67 DATA 236,141,12,20,96,144
    ,1,96,32,140,7,169,10,133,6,
    169
00  68 DATA 0,133,7,32,200,7,32,
    140,7,169,255,141,1,3,230,7
F2  69 DATA 32,200,7,76,5,200,66
    ,76,65,70,75,160,160,160,196
    ,160
00  70 DATA 160,180,100,160,196,
    160,160,160,66,87,160,96,65,
    160,160,160
09  71 DATA 160,180,2,174,0,20,2
    32,32,160,5,136,200,246,236,
    34,76
15  72 DATA 189,5,160,2,174,0,20
    ,202,32,160,5,136,200,240,96
    ,138
A7  73 DATA 41,3,133,107,173,0,2
    0,41,252,5,107,141,0,20,169,
    4
E7  74 DATA 133,107,102,0,202,20
    8,253,196,107,200,249,96,165
    ,34,32,75
66  75 DATA 242,130,10,10,10,10,
    10,133,00,173,0,20,41,150,5,
    96
45  76 DATA 141,0,20,00,173,12,2
    0,41,31,9,192,141,12,20,169,
    255
50  77 DATA 141,3,20,141,1,20,00
    ,165,34,32,75,242,133,67,32,
    213
01  78 DATA 5,169,255,141,1,20,1
    69,0,133,180,170,168,165,57
    153,0
6A  79 DATA 3,165,180,153,2,3,16
    5,34,153,3,3,165,19,153,4,3
03  96 DATA 165,10,153,5,3,169,1
    5,153,6,3,153,7,3,169,0,00
40  81 DATA 2,3,00,3,169,4,3,69
    ,5,3,153,1,3,24,152
7A  82 DATA 105,0,168,230,180,16
    5,180,197,67,144,193,152,72,
    232,130,157
26  83 DATA 0,4,232,200,250,169,
    75,141,0,4,169,3,133,45,32,4
    0
7F  84 DATA 254,104,168,136,32,2
    29,253,32,245,253,169,4,133,
    45,32,233
4F  85 DATA 245,133,96,32,143,24
    7,169,0,133,96,169,255,141,1
    ,20,162
41  86 DATA 5,96,254,104,202,208
    ,250,162,10,164,50,00,254,10
    4,105,0
0F  87 DATA 3,141,1,20,200,202,2
    00,243,132,50,162,0,00,254,1
    04,169
2F  96 DATA 65,141,1,20,202,208,
    245,169,255,162,5,96,254,104
    ,141,1
43  89 DATA 20,202,208,247,162,1
    07,96,254,104,180,0,1,141,1,
    20,232
D7  96 DATA 208,244,196,0,96,254
    ,184,177,40,141,1,20,200,206
    ,245,169
97  91 DATA 05,162,0,96,254,184,
    141,1,20,202,208,247,196,180
    ,200,154
C2  92 DATA 96,254,184,00,254,10
    4,76,0,7,32,0,254,105,152,20
    0,3
09  93 DATA 76,103,7,169,200,133
    ,105,165,67,133,100,169,0,13
    3,50,32
60  94 DATA 119,7,162,10,164,50,
    00,254,184,173,1,20,217,0,3,
    200
9D  95 DATA 40,200,236,96,202,20
    8,235,32,119,7,160,107,96,25
    4,104,173
62  96 DATA 1,20,217,0,1,200,20,
    200,206,242,162,252,00,254,1
    04,173
43  97 DATA 1,20,200,48,200,11,2
    00,202,206,242,196,196,208,1
    93,76,103
E0  90 DATA 7,100,100,200,170,19
    0,106,240,3,76,0,6,169,3,56,
    96
11  99 DATA 165,34,201,35,240,0,
    32,130,5,76,0,6,169,1,24,96
A2  196 DATA 169,256,141,5,24,16
    9,3,44,5,24,16,12,44,0,20,40
    
78  101 DATA 246,173,1,20,104,16
    0,0,96,104,104,76,00,7,32,19
    0,7
3F  195 DATA 108,3,133,110,32,10
    3,230,196,27,165,103,5,153,1
    44,3,96
8D  103 DATA 16,247,169,65,141,2
    ,3,169,42,141,3,3,169,17,141
    ,72
F7  104 DATA 3,169,252,141,73,3,
    96,160,0,152,153,0,3,200,200
    ,250
4A  105 DATA 00,165,144,133,0,16
    5,0,48,252,96,234,234,234,23
    4,156,0
57  196 DATA 105,32,207,240,6,32
    ,22,231,200,208,245,160,105,
    59,207,240
78  107 DATA 6,32,22,231,200,208
    ,245,32,19,206,196,0,165,74,
    207,240
5A  196 DATA 6,32,22,231,200,200
    ,245,32,70,200,32,9,207,160
    ,132
3D  100 DATA 212,165,96,207,240,
    6,153,240,4,200,200,245,32,3
    202,196
CB  110 DATA 0,185,120,207,240,6
    ,153,240,4,200,200,245,32,24
    3,206,200
FF  111 DATA 3,32,123,206,160,0,
    185,103,207,240,6,153,240,4,
    200,206
CA  112 DATA 245,32,243,206,200,
    19,32,9,207,234,234,234,160,
    0,132,212
3A  113 DATA 76,126,205,136,16,2
    40,76,120,205,169,147,32,22,
    231,96,234
75  114 DATA 234,234,234,234,234
    ,234,234,234,234,234,234,234
    ,234,234,234
0A  115 DATA 234,234,234,169,1,1
    33,212,160,0,32,207,255,153,
    144,207,200
02  116 DATA 201,13,208,245,192,
    10,144,5,32,114,206,240,230,
    136,152,240
C5  117 DATA 232,208,169,160,153
    ,143,207,192,10,144,245,136,
    185,144,207,153
20  118 DATA 54,203,136,16,247,9,
    6,169,1,133,212,160,0,32,207
    ,255,153
04  119 DATA 144,207,200,201,13,
    208,245,192,3,240,6,32,110,2
    06,76,78
3F  120 DATA 208,173,144,207,141
    ,72,203,173,145,207,141,73,2
    03,96,132,1
00  121 DATA 248,251,169,20,32,2
    2,231,136,200,248,96,169,0,1
    33,212,169
74  122 DATA 147,32,210,255,234,
    234,234,234,169,1,133,103,16
    9,207,133,100
54  123 DATA 165,134,133,107,169
    ,96,133,105,32,213,243,165,1
    96,32,100,255
00  124 DATA 165,105,32,150,255,
    169,0,133,144,160,3,132,189
    32,165,255
01  125 DATA 133,195,32,165,255
    133,196,104,144,200,53,164,1
    03,136,200,235
C5  126 DATA 160,0,189,32,32,22
    231,136,200,250,196,195,169,
    196,32,205
GC  127 DATA 109,169,32,32,22,23
    1,32,165,255,166,144,256,19,
    201,0,240
F2  128 DATA 6,32,22,231,76,214,
    205,169,13,32,22,231,160,2,2
    00,107
47  129 DATA 76,66,246,32,220,25
    5,200,251,32,220,255,240,251
    ,201,00,200
3A  130 DATA 1,96,201,70,200,237
    ,169,1,96,234,234,234,234,23
    4,234,234
3C  131 DATA 234,234,234,234,234
    ,234,234,234,234,234,234,234
    ,234,234,96,234
13  132 DATA 147,13,32,70,79,02,
    77,65,04,04,69,02,32,79,00,6
    9
04  133 DATA 02,65,04,73,79,70,6
    5,76,32,56,,0,13,32,60,73,03
17  134 DATA 75,32,70,65,77,65,3
    2,42,32,0,13,32,60,73,03,75
D3  135 DATA 32,73,46,00,46,42,3
    2,0,32,6,15,10,13,1,20,20
F9  136 DATA 9,14,7,46,46,32,0,3
    2,1,14,15,20,8,5,10,32
0C  137 DATA 4,9,15,11,32,63,32,
    0,32,3,1,20,1,12,15,7
30  138 DATA 21,5,32,63,32,0,36,
    14,254,246,83,0,0,0,0,0
90  139 DATA 66,87,13,160,160,16
    0,160,160,160,160,160,160,16
    0,160,160,160
E2  140 DATA 160,160,0,255,0,2,0
    ,16,255
```

MULTIFILE 64

A database that can easily be tailored to your own needs
For C64 plus disk drive

Multifile is a disk-based data filing program. It has been constructed to offer the routines necessary for such a program, such as input and retrieval of data, as well as printing out neatly in columns. However it has been designed to allow it to be easily tailored to suit your own requirements. It could easily be converted to handle names and addresses (for personal use or club records), a collection catalogue, a stock control index, in fact the possibilities are endless.

Storing large amounts of array data in Basic introduces large time delays in array handling, so the main data storage routines of this program have been written in machine code. However, the main program is in Basic for ease of customisation for your own use.

Multifile comprises of two programs, the first is a Basic loader program for the machine code section this loads the machine code (in a series of DATA statements) into memory at 49152. When the data is correct the machine code section (just under 1k) is SAVEd as a machine code file to disk. The main program can then load this file directly, and the loader with the machine code data need only be used once, hence saving time when the program is in use.

The second program is the Basic section of the filing system itself, and provides easy access to the machine code routines used.

Data storage

The data is stored by the program in a series of records, each record comprising of a series of data items, about one particular item. Each of these data items is called a field. The data is stored as in Figure 1.

Multifile will handle up to 10 fields, and up to 255 records in a file. Field length is not fixed, but the total length of all fields must not exceed 255 characters (this should not be a problem if a printout of the file on a standard 80-column printer is desired). The number of records in the file is updated as the program is used, but the number of fields required and their lengths must be set up as part of the program before any data files are created. Several versions of the program would required to be kept if the program was used for a variety of filing applications.

Setting up for use

The only lines needing to be changed in the Basic program are lines 500-560 (though the program title in lines 210 and 1000 could be altered also), and these should be changed to your requirements as you enter the listing. Line 500 defines variable F, the number of fields in the files handled. Lines 510-520 define the length (ie. number of characters) of each of the fields (if F is less than 10, the unused field lengths should be set to zero), and lines 530-560 define the titles of each of the fields. These titles are used within the program to refer to the columns, and are also printed as a header on any printout of the file. Note that the length of the title must be the same as the corresponding field size, otherwise an error will occur on running the program. If necessary the field titles should be filled out with spaces eg. if FL(2)= 10, then FT$(2)=" SURNAME". With these variables defined correctly the program is ready to run.

The memory used by the program is as follows: 6 bytes + F bytes (F=number of fields) + record storage. Each record uses memory as follows: 2 bytes + F bytes + 1 byte for each character in each field. All records are stored as string (ie character) data, including any numbers stored. Memory is hence used more efficiently (and more quickly) than from Basic.

Data is stored upward from location 20000 ($4E20), giving a maximum storage capacity of just over 20k per file. Note that records are not necessarily stored in memory in numerical order though this is transparent to the user ie they always appear to be in order when listed. Data is saved as a block of program memory from 20000 to the end of the file.

Using the program

Upon RUNning the program the machine code section is loaded in from disk if it is not already in memory. A menu of functions is then presented, and offers the following options (note that if entry to options is made in error, pressing RETURN from most prompts causes a return to the previous menu).

View Data in File

After choosing the output device (screen or printer (device 4)), or exit back to the menu, all of the data file currently in memory is listed out. The field titles are printed at the top, with the fields neatly printed out in vertical columns beneath their headings with one space between each column. The number of each record is printed down the left side. Listing can be slowed by pressing the CTRL key, and can be paused completely by holding SHIFT or clicking SHIFT LOCK. Pause is indicated by a red border, and the listing will continue when SHIFT is released.

IAIN MURRAY

HOW IT WORKS.

10–410	Titles and load machine code
500–590	Set up data lengths and check
600–620	Set up file memory area
700–790	Set up addresses of machine code routines
1000–1120	Print main menu and get selection
1500–1570	Print out data to screen or printer
2000–2140	Add record to file
2500–2590	Change record in file
3000–3080	Delete record from file
3500–3990	Disk file handling procedures
4000–4390	Data processing routines.
4500–4540	Exit program

Subroutines

6000–6080	Input new or modified record
7000–7070	Get disk filename and store in memory
8000–8030	Read record from machine code buffer
8500–8620	Check for empty file memory
9000–9050	Place record into machine code buffer
9500–9700	Wait for SPACE to be pressed
9800–9850	Check disk drive error channel and report

Add a Record to File.
If more than one record is in the file already the program asks for the number of the record after which the current one is to be added. Each of the field titles is then given, and an input is requested for this field. The whole record is then stored in memory by the machine code routine.

Change Record in File.
The program prompts for the number of the record to be changed, and withdraws this record from memory. Each field title is then given along with the current entry in this field, and a new entry is prompted for. If no change is required to this entry then pressing RETURN on it's own will indicate this. When all fields have been done, the old record is deleted from memory, and the new one added.

Delete Record from File.
The program prompts for the number of the record to be deleted, and it is deleted from memory and printed on the screen.

Disk File Handling.
Selecting this option presents another menu, offering file handling options as follows.

Load Data File
A file name is requested (the FILE extension should not be given), then the required file is loaded from disk. If a file transfer error occurs, this is indicated, otherwise a successful LOAD is indicated. After LOADing a check is made to see if the file is compatible with the field lengths set up within the program, and if non-compatibility is found, a warning is given on the screen.

Save Data File
A file name is requested, then the data file currently in memory is SAVEd to disk. If a file transfer occurs this is indicated, otherwise a successful SAVE is indicated. Note that the extension ,FILE is added to the given file name to indicate in the disk directory that this is a data file.

Disk Directory
A directory of the current disk is displayed on the screen. Pressing CTRL will slow the listing, and pressing SHIFT will pause the listing.

Rename a File
The file's current name is requested, followed by it's new one, the file is then renamed on the disk.

Delete a File
The file's name is requested, and the file deleted from the disk.

Return to Main Menu
The program returns immediately to the main menu screen.

Process Data.
Selecting this option presents another menu, offering data processing options as follows.

Search for Data
A search string of between 2 and 40 characters is asked for, the memory is scanned for all occurrences of this string. All records containing the search string are printed out in full along with their numbers, and a total number of finds is printed.

Sum column
A menu of field titles is printed, and a prompt for one of these is given. All elements in the requested column are then added up, and a total and average for the column are printed. This process may take some time for a long data file.

Return to Main Menu
The program returns immediately to the main menu screen.

Exit Program
After asking for confirmation of exit, the program exits back to Basic.

Figure 1

	(No.)	FIELDS NAME	ADDRESS	PHONE	MEMBER
R	001	J.Adams	3,Main St.	123456	654
E			Anytown		
C	002	F.Jones	10,High St.	987654	321
O			Uptown		
R	003	G.Smith	54 New St.	888666	432
D			Downtown		
S	004	P.Young	17,Low Rd.	765765	223
			Anytown		

```
MULTIFILE MC GEN
```

```
41   10 REM *** MULTIFILE MACHINE
      CODE GENERATOR ***
AE   20 REM *** BY IAIN MURRAY (C
     ) 1986 ***
08   30 REM *** FOR YOUR COMMODOR
     E ***
34   50 POKE 53280,6 POKE 53281,1
     1
1C   60 PRINT "{CLR BLACK DOWN3 R
     IGHT3 RVSON} MULTIFILE MACH
     INE CODE GENERATOR {RVSOFF}
     "
10   70 PRINT "{DOWN3 RIGHT3}THIS
      WILL SAVE  FILECODE  TO DIS
     K"
7F   80 PRINT "{DOWN5 RIGHT7}LOAD
     ING DATA - PLEASE WAIT"
E3   100 AD=49152 C=0
11   110 READ A POKE AD,A C=C+A A
     D=AD+1
04   120 IF AD<50135 THEN 110
BE   130 IF A=195 AND C=130109 TH
     EN 150
9A   140 PRINT "{DOWN3 RIGHT7}ERR
     OR IN DATA    END"
41   150 PRINT "{DOWN3 RIGHT5}DAT
     A OK - PRESS {RVSON} SPACE {
     RVSOFF} TO SAVE"
0A   160 GET A$:IF A$<>" " THEN 1
     60
E9   170 POKE 43,0:POKE 44,192:PO
     KE 45,220:POKE 46,195
D0   180 SAVE "FILECODE",8
3E   190 END
64   1000 DATA 165,251,56,233,1,1
     33,251,165,252,233
70   1010 DATA 0,133,252,96,165,2
     51,24,105,1,133
0E   1020 DATA 251,165,252,105,0
     ,133,252,96,169,44
EC   1030 DATA 133,251,169,78,133
     ,252,96,160,0,177
51   1040 DATA 251,201,255,208,7,
     200,177,251,201,255
33   1050 DATA 240,6,32,14,192,76
     ,37,192,200,177
61   1060 DATA 251,96,173,2,206,2
     00,12,160,255,169
83   1070 DATA 1,141,3,206,169,3,
     76,03,192,160
83   1080 DATA 7,169,4,17D,32,186
     ,255,169,0,32
D9   1090 DATA 189,255,32,192,255
     ,169,3,24,109,2
38   1100 DATA 206,170,32,201,255
     ,76,101,193,32,231
74   1110 DATA 255,169,D,141,3,2D
     ,6,96,169,3,141
1A   1120 DATA 5,206,24,109,32,70
     ,141,4,206,169
41   1130 DATA 0,141,6,206,172,4
     ,206,177,251,32
C0   1140 DATA 210,255,238,4,206
     ,238,6,206,172,5
70   1150 DATA 206,177,251,205,6
     ,206,208,232,172,5
CF   1160 DATA 206,165,31,70,56,2
     33,1,205,6,206
51   1170 DATA 48,20,152,56,233,2
     ,205,32,70,240
6A   1180 DATA 11,169,32,32,210,2
     55,238,6,206,76
92   1190 DATA 158,192,169,32,32,
     210,255,238,5,2D6
82   1200 DATA 173,5,206,56,233,3
     ,205,32,70,2D0
BA   1210 DATA 174,169,13,32,210
     ,255,32,21D,255,96
57   1220 DATA 255,255,255,32,62,
     192,240,1,96,169,1
CD   1230 DATA 141,7,2D6,32,94,19
     5,32,20,192,32
40   1240 DATA 37,192,201,255,24D
     ,46,2D5,7,2D6,24D
A3   125D DATA 6,32,14,192,76,239
     ,192,32,41,195
D6   1260 DATA 32,6D,195,32,117,1
     92,238,7,206,173
5D   1270 DATA 142,2,2D1,1,2D0,0
     ,169,2,141,32
0E   1280 DATA 2D8,70,13,193,169
     ,6,141,32,208,76
ED   129D DATA 236,192,76,108,192
     ,255,D,169,0,170
93   1300 DATA 160,255,32,186,255
     ,173,0,2D7,162,1
51   1310 DATA 18D,2D7,32,189,255
     ,32,20,192,32,37
92   132D DATA 192,2D1,255,240,6,
     32,14,192,76,62
0F   133D DATA 193,32,14,192,32,1
     4,192,32,14,192
53   1340 DATA 166,251,164,252,16
     9,31,133,251,169,78
F7   135D DATA 133,252,169,251,32
     ,216,255,32,103,255
25   1360 DATA 41,120,141,8,206,9
     6,255,255,169,0
E4   137D DATA 170,160,255,32,186
     ,255,173,0,2D7,162
A9   1380 DATA 1,160,207,32,189,2
     55,169,0,32,213
DA   1390 DATA 255,76,101,193,255
     ,255,32,20,192,32
0F   1400 DATA 37,192,201,255,240
     ,24,205,1,206,240
47   1410 DATA 2,16,6,32,14,192,7
     6,143,193,24
A7   1420 DATA 105,1,145,251,32,1
     4,192,76,143,193
FF   143D DATA 160,0,32,14,192,32
     ,14,192,173,1
87   1440 DATA 206,24,105,1,145,2
     51,32,14,192,185
DA   145D DATA 0,2D5,145,251,20D
     ,200,0,206,20D,245
E8   1460 DATA 169,255,145,251,20
     0,145,251,20D,145,251
ED   147D DATA 238,33,78,96,255,2
     55,32,20,192,32
D0   1480 DATA 37,192,205,9,2D6,2
     4D,6,32,14,192
2C   149D DATA 76,223,193,169,0,1
     41,2,206,32,62
5A   1500 DATA 192,32,117,192,32,
     100,192,174,32,70
27   1510 DATA 169,3,168,24,1D9,3
     2,70,24,113,251
68   1520 DATA 200,2D2,208,249,14
     ,11,206,169,D,141
B9   1530 DATA 1D,206,172,11,206
     ,177,251,16D,0,145
E5   1540 DATA 251,201,255,240,9
     ,140,10,206,32,14
06   1550 DATA 192,76,20,194,238,
     10,206,173,10,206
C2   1560 DATA 201,3,240,6,32,14,
     192,76,20,194
AF   1570 DATA 32,28,192,32,37,19
     2,2D1,255,240,27
0E   1580 DATA 56,237,9,206,176,6
     ,32,14,192,76
9D   1590 DATA 61,194,160,2,177,2
     51,56,233,1,145
44   1600 DATA 251,32,14,192,76,6
     ,1,194,206,33,70
EF   161D DATA 96,0,255,169,48,13
     3,252,169,2,133
79   1620 DATA 253,169,0,133,144,
     169,36,133,251,169
0D   1630 DATA 251,133,187,169,0
     ,133,190,165,253,133
0E   1640 DATA 103,169,8,133,186,
     169,96,133,185,32
4A   1650 DATA 213,243,165,186,32
     ,180,255,165,185,32
3F   1660 DATA 150,255,164,144,20
     ,8,70,160,6,132,251
DD   1670 DATA 32,165,255,166,252
     ,133,252,164,144,208
F3   1680 DATA 55,164,251,136,200
     ,230,164,252,32,205
E6   169D DATA 189,169,32,32,210,
     255,32,165,255,72
17   1700 DATA 173,142,2,201,1,24
     0,249,104,166,144
63   1710 DATA 208,24,170,240,6,3
     2,210,255,76,104
90   1720 DATA 194,169,13,32,210,
     255,165,197,201,63
DF   1730 DATA 240,4,160,4,208,10
     ,8,32,66,246,96
92   174D DATA 255,255,32,28,192,
     173,2,207,24,105
F2   1750 DATA 2,141,2,207,32,37,
     192,205,15,208
A4   1760 DATA 240,6,32,14,192,76
     ,242,194,173,32
C8   1770 DATA 70,170,160,3,204,2
     ,207,208,3,141
0F   1780 DATA 3,207,24,113,251,2
     00,202,208,241,24
44   1790 DATA 105,3,141,0,206,16
     0,3,177,251,153
9F   1800 DATA 253,204,200,204,0,
     206,208,245,96,160
5E   1810 DATA 0,185,12,206,32,21
     0,255,200,192,3
C3   1820 DATA 208,245,169,32,32,
     210,255,96,238,14
47   1830 DATA 206,173,14,206,201
     ,58,208,23,169,48
02   1840 DATA 141,14,206,238,13,
     206,173,13,206,201
AD   1850 DATA 58,208,8,169,48,14
     1,13,206,238,12
20   1860 DATA 206,96,169,48,141,
     12,206,141,13,206
31   1870 DATA 141,14,206,238,14,
     206,96,0,255,32
88   1880 DATA 94,195,162,0,142,4
     ,207,232,142,7
CF   1890 DATA 206,32,28,192,32,3
     7,192,201,255,208
AA   1900 DATA 1,96,205,7,206,240
     ,6,32,14,192
4E   1910 DATA 76,126,195,32,14,1
```

```
    $2 160 0 32 14
D4  1$20 DATA 1$2 177 251 217 61
     3 240 13 201 255
B4  1$30 DATA 208 240 32 6D 1$5,
    238 7 206 76 126
64  1$40 DATA 1$5 200 204 60 3 2
    08 230 32 0 1$2
FD  1$50 DATA 160 0 177 251 201
    255 208 245,2DD 177
59  1$60 DATA 251 201 255 208 23
    8 32 41 195 32 117
6A  1$70 DATA 1$2 32 60 1$5,238
    7 206 238 4 207
7E  1$80 DATA 76 123 195
```

```
PROGRAM  MULTIFILE
```

```
60   10 REM *** MULTIFILE-CONVERT
     [BLE FILE PROCESSOR ***
AB   20 REM *** BY [A1N MURRAY [C
     1 1$86 ***
08   30 REM *** FDR YOUR COMMODOR
     E ***
17   4D REM *** DISK-BASED FILE S
     TORAGE ***
$3   BO UF$=CHR$[147]+CHR$[142] D
     N$=CHR$([147]+CHR$[14]+CHR$[8
     ) REM * CASES
21   100 PRINT  [BLACK] +DN$ REM
     * BLACK/LOWER CASE
1E   110 POKE 53280 6 POKE 53281
     1 REM * SCREEN COLOURS
AF   200 IF PEEK[4$]52]=165 THEN
     300
8$   210 PRINT DN$+"|DOWN2 RIGNT]
     D RVSON|   |SM SU SL ST S] SP
     SI SL SE|
6A   220 PRINT "|DOWN4 RIDHT6 SL
     SO 5A SD S] SN SG SSPC SM SA
     SC SN SI SN SE SSPC SC SO S
     D SE SSPC S$ SE SC ST S] SO
     SN|
D6   230 PRINT (DOWN3 RIGHT]3 RVS
     DN| |SP SL SE 5A S$ SE| |SW
     SA S] ST| |RVSOFF|
6F   240 LOAD  FILECODE  8,1
1C   250 POKE 56,78 REM * MEMORY
     TOP
A0   300 DIM FL[10] FT$[10] FS$(1
     0)
18   400 MM = 20D00 REM * START O
     P DATA AREA
$F   410 BF=52480 REM * M/C INTER
     FACE BUFFER
D7   5D0 F=*** REM * NUMBER OF FI
     ELD$
DE   510 FL[1]=***:FL[2]=*** FL[3
     ]=*** FL(4)=*** FL[5]=*** RE
     M * LENGTH OF FIELD$
CB   520 FL[6]=*** FL(7)=*** FL(8
     )=*** FL[$]=*** FL[10)=*** R
     EM * LENGTH OF FIELDS
67   530 FT$[1]= |ST S] ST SL SE|
      1" FT$[2]="|ST S] ST SL SE|
      2" FT$[3]= |ST S] ST SL SE
     SSPC|3  REM * FIELD TITLES
1F   540 FT$[4]= |ST S] ST SL SE|
      4  FT$[5]= |ST S] ST SL SE|
      5" FT$[6]="|ST.S] ST $L SE|
     $" REM * FIELD TITLES
03   55D FT$[7]= |ST S] ST SL SE
     SSPC17  FT$[8]= |ST S].ST SL
     .SE SSPC|8  FT$[9]="|ST S] S
     T SL SE SSPC)$" REM * FIELD
     TITLES
B1   58D FT$(1D]="|ST.S] ST SL SE
     SSPC|1D" REM * FIELD TITLES

5E   565 IF F<2 OR F>10 OR F<>INT
     (F) THEN PRINT "|SP|]ELD NUM
     BER ERNOR" END
$C   570 FOR I=1 TO F IF LEN[FT$[
     ]]<>FL[1] THEN PRINT  |SSIT
     RING  1  LENGTH ERNOR  END
86   580 FS$[1]=FT$[1] IF LEN(FS$
     [1]}>12 THEN FS0([]=LEFT$[FS
     $[[] 12]
D2   58D NEXT
D7   60D POKE MM,F POKE MM+1 0 RE
     M * SET UP MEMORY AREA
PB   610 FOR I=1 TO F POKE MM+1+I
     ,FL[1] NEXT
3C   620 FOR I=MM+12 TO MM+14:POK
     E I 255 NEXT
E3   700 FT=49374 REM *** PRINTOU
     T ***
F5   710 SV=49449 REM *** SAVE FI
     LE ***
BF   720 LD=49520 REM *** LOAD FI
     LE ***
0A   730 AR=49548 REM *** ADD REC
     ORD ***
0D   740 DR=49628 REM *** DELETE
     RECORD ***
07   750 DY=49765 REM *** DISK DI
     RECTORY ***
C7   760 SR=49894 REM *** RECORD
     SEARCH ***
0B   770 SH=50D31 REM *** DATA SE
     ARCH ***
4O   775 REM *** END OF CODE AT 5
     0130 ***
CB   780 FT$= |DOWN R|GHT7 RVSON|
      FILE TRANSFER COMPLETED |RV
     SOFF|"
4$   790 FE$="|DOWN RIGHT9|FILE T
     RANSFER ERROR!"
0D   8$$ REM *** MAIN MENU ***
F4   1D00 PRINT DN$+ |DOWN2 RIGHT
     13 RVSON|   |SM SU SL ST SI S
     F $I SL SE|
4E   1D10 PRINT  |DOWN RIGHT|3 RV
     SON|   |SM SA SI SNI |SM SE S
     N SU|
32   1020 PRINT  |DOWN2 RIGHT8|1]
     |SSPC SV|)EW |SO|ATA IN |SF|
     ILE"
9C   103D PRINT  |DOWN RIGHT8|2]|
     SSPC SA|DD |SR|ECORD TO |SF|
     ILE
BA   104D PRINT  |DOWN RIGHT8|3)|
     SSPC SC|HANGE |SR|ECORD IN |
     SF|ILE"
5$   1050 PRINT  |DOWN RIGHT8|4)
     |SD|ELETE |SR|ECORD FROM |SF
     |ILE
0C   1060 PRINT  |DOWN RIGHT8|5)
     |SD|ISK |SF|ILE |SH|ANDLING

9E   1070 PRINT "|DOWN RIGHT8|6]
     |SP|ROCESS |SO|ATA
DE   1080 PRINT  |DOWN RIGHT8|7)
     |SE|XIT |SF|ROGRAM
54   1100 PRINT  |DOWN2 RIGHT4 SW
     ]HICH DO YOU REQUIRE [1 71 ?
F2   1110 GET A$ A=VAL[A$] IF A<1
      OR A>8 THEN 1]10
A4   1120 ON A GOTO 1500 2000 250
     0 3000 3500 4000 4500
6A   1499 REM *** VIEW FILE DATA
     ***
54   1500 PRINT  |CLR DOWN2 RIGHT
     11 RVSON SV]IEW |SDIATA IN |
     SF|ILE
F2   1505 GOSUB 8500 IF MX=0 THEN
      1000
CE   1510 PRINT  |DOWN2 RIGNT4 SO
     |UTPUT TO |SS|CREEN OR |SPIR
     INTER  OR
7B   1520 PRINT  |RIGHT4|RETURN T
     O |SH|A]N |SM|ENU [|SS|/|SP|
     /|SM|)|SSPC|?
BA   1530 GET A$ IF A$= M  THEN 1
     DD0
53   1534 DV=0 IF A$= P  THEN DV=
     1 GOTO 1530
31   1536 IF A$<> S  THEN 1530
4C   1538 POKE 52730,DV
37   1540 PRINT  |DOWN2 RIGNT6 SP
     |RESS |RVSON S8 SH S1 SF ST
     RVSOFF| TO PAUSE LISTING|DOW
     N2|"
0D   1550 IF DV=1 THEN PRINT  |RI
     GHT15 RVSON| |SP 5R S] SN ST
      $I 8N SG| |RVSOFF DOWN|  OP
     EN 1 4 7 GOTO 1555
F6   1553 OPEN 1 3
47   1555 PRINT#1," 0  " FOR I=1
     TO P:PRINT#1,"  FT$(I)  NEX
     T PRINT#1 PRINT#1 CLOSE]
1A   1560 SYS FT IF PEEK[52744|<>
     0 THEN PRINT  |UP2 RIGHT6 SP
      SR S] SN ST SE SR SSPC SN 5
     O ST SSPC 5A SV 8A S] SL 5A
     S0 SL SE|'
32   1570 GOSUB $600 GOTO 1000
0D   1$$$ REM *** ADD DATA TO FIL
     E ***
63   20D0 PRINT  |CLR DOWN2 RIGHT
     12 RVSON 8A]DD |SDIATA TO |S
     F|ILE|DOWN2|
31   2D10 PRINT  |DOWN2 RIGHT4 SD
     )O YOU WISH TO ADD A RECOAD
     [|SY|/|SN|) ?
27   2020 GET A$ IF A$= N  THEN 1
     ODD
1$   2030 IF A$<> Y  THEN 2020
0F   2040 MX=PEEK(MM+1)-1 IF MX=0 T
     HEN MX=1
$1   2045 IF MX=1 THEN A=1 DOTO 2
     D80
72   2050 PRINT  |DOWN2 RIGNT4 SW
     |HICH RECORD DO YOU WISH TO
     RDD
80   2060 PRINT  |RIGHT4|AFTER [
     1 -" MX "]  |8SPC| "
C5   2070 INPUT A$ A=VAL[A$] IF A
     <1 OR A>MX THEN 2050
BF   2080 POKE 52737 A IF PEEK(MM
     +1)=0 THEN POKE 52737 0
1D   2120 RF=0 GOSUB 6000
F7   2130 SYS AR
13   2140 PRINT  |DOWN2 RIGHT13 S
     RIECORD |SA]DDED  GOSUB $600
      GOTO 1000
CD   2499 REM *** CHANGE A RECORD
     ***
EB   2500 PRINT  |CLR DOWN2 RIGHT
     13. RVSON SC]HANGE |SR|ECORD
```

```
59  2510 GOSUB 8500 IF MX=0 THEN
    1000
AE  2520 PRINT "{DOWN2 RIGHT4 SW
    }HICH RECORD DO YOU WISH TO
    CHANGE"
E3  2530 PRINT "{8}{RIGHT4}( 1 -" M
    X")"
04  2540 A$="":INPUT A$ IF LEN(A
    $)=0 THEN 1000
26  2550 A=VAL(A$) IF A<1 OR A>M
    X THEN 2520
CA  2560 POKE 52751,A SYS SR GOS
    UB 9000 RP=1 GOSUB 6000
AC  2565 PRINT "{WHITE}" REM * 
    WH(TE FOR INV(S)E)LITY
3E  2570 POKE 52745,A SYS DR REM
     * REMOVE OLD RECORD
26  25RO POKE 52737,A:1 SYS AR:R
    EM * ADD NEW RECORD
30  2590 PRINT "{BLACK UP RIGHT1
    0 SR}ECORD |SC|HANGED" GOSUB
     9600 GOTO 1000
04  2999 REM *** DELETE A RECORD
     ***
74  3000 PRINT "{CLR DOWN2 RIGHT
    13 RVSON SD}ELETE |SR|ECORD"
1F  3010 GOSUB 8500 IF MX=0 THEN
    1000
31  3020 PRINT "{DOWN2 RIGHT4 SW
    }N|CN RECORD OO YOU WISN TO
    DELETE"
59  3030 PRINT "{RIGHT4}( 1 -" M
    X")"
AA  3040 A$="":INPUT A$ IF LEN(A
    $)=0 THEN 1000
06  3050 A=VAL(A$) IF A<1 OR A>M
    X THEN 3020
1D  3060 PRINT "{DOWN2}" POKE 52
    745,A SYS DR
B5  3070 PRINT "{DOWN RIGNT12 SR
    |ECORD |SD|ELETED"
B6  3080 GOSUR 9600 GOTO 1000
65  3499 REM *** DISK FILE HANDL
    ING ***
5F  3500 PRINT DN$+ "{DOWN2 RIGHT
    II RVSON SD|I|SK |SF|ILE |SH|
    ANDLING|DOWN2|"
10  3510 PRINT "{RIGHTR}1I |SL|O
    AD |SD|ATA |SF|ILE"
55  3520 PRINT "{DOWN RIGHT8}2|
    |SS|AVE |SD|ATA |SF|ILE"
F3  3530 PRINT "{DOWN RIGHT8}3|
    |SD||SK |SD||RECTORY"
6F  3540 PRINT "{DOWN RIGNT8}4|
    |SR|ENAME A |SF||LE"
DF  3550 PRINT "{DOWN RIGHT8}5|
    |SD|ELETE A |SF||LE"
F3  3560 PRINT "{DOWN RIGNT8}6|
    |SR|ETURN TO |SW|AIN (SM)ENU
    "
E2  3570 PRINT "{DOWN2 RIGHT4 SW
    }NICN DO YOU REQUIRE (1 6) ?
    "
3E  3580 GET A$ IF A$< 1 OR A$>
    6 THEN 3580
88  3590 A=VAL(A$) ON A GOTO 370
    0 3800 3600 3900 3950 1000
58  3600 PRINT UP$+"{DOWN2 RIGHT
    4|PRESS |RVSON|SN|FT|RVSOFF}
    TO PAUSE DIRECTORY{DOWN2}"
D0  3610 SYS DY GOSUB 9500 GOTO
    3500
AB  3700 PRINT UP$+ "{DOWN2,RIGHT
    15 RVSON|LOAD DATA"
A0  3710 GOSUB 7000 IF FM=0 THER
    3500
56  3740 GOSUB 9700 PRINT "{DOWN
    ,RIGHT15 RVSON} LOADING |RVS
    OFF}"
4A  3750 SYS LD
40  3760 IF PEEK(52744)<>0 THEN
    PRINT FE$ GOTO 3780
D0  3770 GOSUP 9R00 IP ER<20 THE
    M PRINT FT$
41  3775 IF ER>-20 THEN 3780
P9  3776 F=PEEK(MM) FOR I=1 TO F
    FL(I)=PEEK(MM+I+1)
62  3777 IF FL(1)<>LEN(FT$(1)) T
    HEN PRINT "{DOWN RIGHT10}INC
    OMPATI@LE FILE" GOTO 3780
52  3778 NEXT
68  3780 GOSUB 9500 GOTO 3500
3E  3800 PRINT UP$+ "{DOWN2 RIGHT
    15 RVSON|SAVE DATA"
2C  3810 GOSUB 8600 IF MX=0 THEN
    3500
C1  3820 GOSUB 7000 IF FM=0 THEN
    3500
27  3R40 GOSUB 9700 PRINT "{DOWN
    ,RIGHT15 RVSON} SAVING |RVSO
    FF}"
4A  3850 SYS SV
36  3860 GOTO 3760
BF  3900 PRINT UP$+ "{DOWN2 RIGHT
    14 RVSON|RENAME FILE"
A7  3910 PRINT "{DOWN2 RIGHT4}GI
    VE PRESEHT FILENAME"
81  3920 PRINT "{DOWN SPC6}12345
    67890|23456"
FE  3925 INPUT "}SPC4}" CN$ FM=
    CN$ IF LEN(CN$)=0 THEN 3500
88  3927 IF LEN(CN$)>16 THEN 391
    0
E2  3930 PRINT "{DOWN2 RIGHT4}GI
    VE NEW FILENAME"
22  3935 PRINT "{DOWN SPC6}12345
    67890|23456"
A7  3940 INPUT "{SPC4}" NN$ IF L
    EN(NN$)=0 THEN 3500
88  3942 IF LEN(NN$)>16 THEN 393
    0
C6  3945 PRINT "{DOWN RIGHT15 RV
    SON} RENAMING |RVSOFF}"
A3  3949 OPEN 1,8,15,"R:"+NN$+"=
    "+CN$:CLOSE 1 GOSUB 9800 GOS
    UB 9500 GOTO 3500
PA  3950 PRINT UP$+ "{DOWN2 RIGHT
    14 RVSON|DELETE FILE"
02  3960 PRINT "{DOWN2 RIGHT4}GI
    VE FILENAME"
4F  3970 PRINT "{DOWN SPC6}}2345
    67890123456"
4D  3980 INPUT "{SPC4}" CN$ FM=
    CN$ IF LEN(CN$)=0 THEN 3500
2A  3982 IF LEN(CN$)>16 THEN 396
    0
30  3985 PRINT "{DOWN RIGHT14 RV
    SON} DELETING |RVSOFF}"
4A  3990 OPEN 1,8,15,"S:"+CN$:CL
    OSE 1 GOSUB 9000 GOSUB 9500
    GOTO 3500
84  3999 REM *** PROCESS DATA **
    *
B6  400D PRINT "{CLR DOWN2 RIGHT
    14 RVSON SP|ROCESS |SD|ATA"
65  4004 GOSUB 8500 IF MX=0 THEN
    1D00
CB  4010 PRINT "{DOWN2 RIGHT4}{1}
    |SS|EARCH FOR |SD|ATA"
A5  4020 PRINT "{DOWN RIGHT4}{2}
    |SS|UM |SD|ATA |SC|OLUMN"
9A  4030 PRINT "{DOWN RIGNT4}{3}
    |SR|ETURN TO |SM|AIN |SM|ENU
    "
8A  4040 PRINT "{DOWN2 RIGHT4 SW
    }NICH DO YOU REQUIRE |1-3| ?
    "
46  4050 GET A$ A=VAL(A$) IF A<1
    OR A>3 THEN 4050
19  4060 PRINT A$ "{DOWN}" ON A
    GOTO 4100 4300 1000
R9  4100 PRINT "{DOWN RIGHT4 SG|
    }VE SEARCH STRING"
C1  4110 INPUT S$ S=LEN(S$) IF S
    <2 OR S>40 THEN 4100
61  4120 PRINT "{DOWN}" POKE 82R
    ,S FOR I=1 TO S POKE 828+I A
    SC(MID$(S$,I,1)) NEXT
E7  4130 SYS SH
77  4140 PRINT "{DOWN RIGNT4 ST|
    OTAL OF PEEK(52996) OCCURR
    ENCES FOUND"
7C  4150 GOSUB 9600 GOTO 1000
C8  4300 FOR I=1 TO F STEP 2 PR|
    NT "{RIGHT3}" I "{LEFT|}" F
    $$(I)
C4  4305 IF I+1>F THEN PRINT GOT
    O 4310
B5  4307 PRINT TAR(20)I+1 "{LEF
    T}|" FS$(I+1)
06  4310 NEXT PRINT "{DOWN RIGHT
    4 SW}HICH COLUMN DO YOU WISH
     TO SUM
AA  4315 PRINT "{RIGHT4}|" 1 "-" F
    "|"
6P  4320 }NPUT A$ A=VAL(A$) IF A
    <1 OR A>F THEN PRINT "{DOWN|
    " GOTO 4300
3C  4330 N=0 PRINT "{DOWN2 RIGHT
    15 RVSON} |SS SU SM2 S| SH S
    G| |RVSOFF|"
90  4340 FOR W=1 TO MX POKE 5275
    1,W POKE 52994,A SYS SR
47  4350 PL=PEEK(RF+A-1) RX=PEEK
    (52995)+RF N$=""
39  4360 FOR J=0 TO RL-1:N$=N$+C
    HR$(PEEK(RX+J)) NEXT
1D  4370 N=N+VAL(N$) NEXT
2B  4380 PRINT "{DOWN2 RIGHT4 SC
    }OLUMN A | FS$(A) "| TOTA
    L = N
7B  4385 PRINT "{DOWN RIGHT4}|8
    T}OTAL OF MX "ITEMS}
5C  4390 PRINT "{DOWN RIGHT4 SC|
    OLUMN AVERAGE = N/MX GOSUB
    9600 GOTO 1000
19  4499 REM *** EXIT PROGRAM **
    *
27  4500 PRINT "{CLR DOWN2 RIGHT
    14 RVSON SE}XIT |SP|ROGRAM|D
    OWN2|"
64  4510 PRINT "{DOWN2 R}IGNT4 SD
    10 YOU W}SH TO EXIT ||SY|/|S
    N|| ?
D0  4520 GET A$ IF A$= N" THEN 1
    000
EE  4530 IF A$<> Y" THEN 4520
56  4540 PRINT UP$+CHR$(9)+ "{DOW
    N20}" END
DE  5999 REM *** GET NEW DATA RE
    CORD ***
F2  6000 FOR I=1 TO F
3B  6020 IF RP=1 THEN PRINT "{DO
    WN RIGHT4 3E}XISTING FS$(I
    )+" IS" PRINT "{RIGHT4}"D
    $(I)
72  6025 IF RP=0 THEN D$(I)=""
```

```
82  6030 A$=  :PRINT "{DOWN RIGH
    T4 SN}EW  FS6{III}"  |SI}TEM
    "
15  6035 INPUT  [8IGHT4|  |A$
7E  6040 IF LEN(A$)<0 THEN 6055
D6  6044 IF LEN(A$)<=FL{II THEN
    6050
4B  6046 PRINT  |DOWN RIGHT4 SD{
    ATA |S}INPUT  |ST}OO |SL}ONG'
     |SSPC}{|SM}UST BE
A8  6047 PRINT  |RIGHT3{  FL{I}
    CHARACTERS LONG OR LESS!" G
    OTO 6020
57  6050 D${II=A$ DL{I}=LEN(A$)
E0  6055 IF LEN(D$(II)=0 THEN PR
    INT  {DOWN RIGHT4 SN}O {SD}A
    TA |SI}NPUT'  GOTO 6020
28  6060 NEXT I
E9  6070 GOSUB 9000
59  6080 RETURN
74  6599 REM *** GET F}LENAME AN
    D STORE ***
00  7000 PRINT  |DOWN2 RIGHT22}}
    234567890{
AF  7010 PRINT "|R}IGHT4|GIVE FIL
    ENAME  |SSPC} ;
88  7020 FM6=  )NPUT FM$ FM=LEN
    (FM$)
A1  7025 IF FM=0 THEN 7070
70  7030 IF FM<2 OR FM>11 THEN 7
    000
73  7040 FM6=FM6+  FILE
69  7050 POKE 52992 FM+5
7C  7060 FOR I=1 TO FM+5 POKE 52
    992+I ASC(MID$(FM$ I 111 NEX
    T
37  7070 RETURN
75  7999 REM *** GET RECORD FROM
     M/C BUFFER ***
9D  8000 PRINT  |DOWN R{IGHT13 RV
    SON} |SP SL SE SA SS SE SSPC
     SW SA SI ST} {RVSOFF}"
F7  8005 FOR I=1 TO F DL{II=PEEK
    (8F+I 1I D$(II=  NEXT
FC  8010 TT=0 FOR I=1 TO F
50  8020 FOR J=1 TO DL{II D$(II=
    D$(II+CHR$(PEEK(8F+F+TT+J-1I
    ) NEXT TT=TT+DL(I} NEXT
F3  8030 RETURN
90  8500 MX=PEEK(MM+I)
2E  8510 {F MX=0 THEN PRINT "|DO
    WN4 RIGHT10 SN SO| )SR SE SC
     SO SR SD SS} |SI SN} |SP SI
     SL SE|  GOSUB 9600
E7  8520 RETURN
3C  8600 MX=PEEK(MM+II
91  8610 IF MX=0 THEN PRINT  |DO
    WN4 R}GNT10{NO RECORDS }N F}
    LE  GOSUB 9500
43  8620 RETURN
B4  8999 REM *** PUT RECORD INTO
     M/C BUFFER ***
59  9000 PRINT "|DOWN R}GNT13 RV
    SON| |SP SL SE SA SS SE SSPC
     SW SA SI ST} |RVSOFF}"
7A  9005 FOR )=1 TO F POKE (8F+)
    -II DL{II NEXT TT=0 DL=F
8C  9010 FOR I=) TO F
B5  9020 FO8 J=1 TO DL{II POKE (
    8F+F+TT+J-1I ASC(MID$(D$(I)
    J 1II
37  9030 NEXT TT=TT+DL{I} 8L=8L+
    DL(II NEXT
EE  9040 POKE 52736 8L REM * 8UF
    FE8 LENGTH
F3  9050 RETURN
F6  9499 REM *** WAIT FOR SPACE
    PRESS ***
6I  9500 PRINT "|DOWN RIGHT7}PRE
    SS |RVSON} SPACE |RVSOFF} TO
     CONTINUE
39  9510 GET A$ {F A$<>"  THEN
    9510
DB  9520 RETURN
2D  9600 PRINT  {DOWN RIGHT7 SP}
    RESS |RVSON} |SS SP SA SC SE
    } |RVSOFF} TO |8C}ONTINUE"
B5  9610 GOTO 9510
2C  9700 PRINT  |DOWN RIGHT11|PR
    EPARE DISK AND  GOTO 9500
CF  9799 REM *** GET DISK STATUS
     ***
D3  9800 OPEN 15 8 15:INPUT#15 E
    R ER6 CLOSE I5
DE  9810 IF ER<20 THEN 9850
20  9815 PRINT  {UP2}  FE$
6E  9820 PRINT  {DOWN RIGHT9|ERR
    OR 6  ER
11  9830 PRINT  |DOWN R|GNT9}(
    ER6  )
50  9840 PRINT  |DOWN R}GHT9|FIL
    ENAME |  FM6
2E  9850 RETURN
```

Disk File Describer

Keep track of which files do what with this handy disk utility

Picture this scenario: you have just spent hours on your epic program and have yet to tie the loose ends, but it's getting late and you'll finish it off the next day. You SAVE your routine with all others, any old file name will do and pack up for the night.

Unfortunately, the next day you're busy and the day after you've just bought a cracker of a game and then spend four months getting an unbeatable score on it. You flick through your old disks 'cos you're running short on space and you find that disk you worked on all that time ago. I wonder what these files are, names like TRY1 and ML4, even a TZXE27!

Your epic has gone hasn't it? Has it? Not now! This little program will help you to remember even the most odd-named files you happened to SAVE with a description of up to 55 characters in length! A SYS for an old game perhaps? No need to remember it now, write it in the file describer and it will never get lost.

The Commodore DOS on the 64 is unfortunately limited in several ways, not least in the fact that file names may only have 16 characters. For most applications it isn't enough, this program allows you to add a definitive description of 55 characters to any file on the disk. Merely create the descriptive titles and SAVE them onto the particular disk to which it refers. Then simply LOAD and RUN Disk File Describer and, if you desire, additions and editions can be made at any time.

Getting it in

Disk File Describer is a Basic program, and thus fairly simple to follow and type in. Use the *SYNTAX CHECKER* program that is found on the *LISTINGS* page to help you with your typing.

After RUNning the program loads the description file if one exists, then LOADs the directory. On screen you are presented with a menu offering five choices.

Option 1 views the directory together with the descriptions and a further menu at the bottom of the screen gives options to add or alter the current text. The file names are shown in blocks of six. If there are more files on disk keys f5 and f7 allow you to page back and forth as required. Pressing the f3 key puts you into edit mode.

Once into the edit mode you enter the file number (the number next to the files on the screen). The cursor then moves to the start of the light field in which you can make the necessary text edition. If you have made an incorrect choice of file then just press RETURN and the directory screen will return with no alterations made. All characters can be entered within the description area except inverted commas (quote marks) and although the cursor travels automatically to the next text line it cannot distinguish ends of words unlike a word processor. If you are a tidy sort of person, you will no doubt space your description so that it is easily readable at a later date. The INST/DEL key erases as per it's normal function and RETURN after completion of entering your desired text brings the directory screen back for you to continue entering descriptive text or, by pressing f1 and option 3, SAVE the description file onto disk. Make sure that enough room exists on the disk for this file.

As far as the other options are concerned, option 2 gives you normal DOS commands such as scratch, validate and rename. It also formats disks.

As stated, option 3 SAVEs the directory descriptions to disk as a sequential file. If a file is already present on disk the program will overwrite it with the current data in memory. Choosing option 4 simply allows you to work with other disks and reRUNs the program as this option is chosen.

To exit the program choose option 5.

I am sure you will find many uses for this program and it would be interesting to hear how it could be adapted to individual needs.

```
 PROGRAM: DISK DESCRIBER

04   1 REM ************************
     *
27   2 REM * DISK FILE DESCRIBER
     *
3B   3 REM *      BEN WELLBAY
     *
9D   4 REM * YOUR COMMODORE 1986
     *
0C   5 REM ************************
     *
CA   6 POKE53280,11:POKE53281,11:
     PRINT"{CLR,DOWN6,RIGHT4,C=}D
     ISK FILE DESCRIBER"
2C   7 PRINT"{DOWN2,RIGHT4}PLEASE
      WAIT - READING DATA"
39   8 FORI=830TO902:READA:POKEI,
     A:NEXT
9C   9 A1$="{HOME,DOWN21}":FL=1
61   10 A2$="{RVSON,SPC19,ENDOFF,
     SPC4,RVSON,SPC36}"
71   11 FA=144:DIMDT$(FA),J$(FA)
     K$(FA),J1$(FA)
CE   12 FORI=0TOFA:DT$(I)=CHR$(32
     )+"{SPC18}":NEXT
FE   13 PRINT"{UP,SPC11}READING D
     IRECTORY":OPEN1,8,0,"$":SY
     S830:CLOSE1
83   14 GOSUB80
6F   15 POKE198,0:PRINT"{CLR,DOWN
     }"SPC(6)"{RVSON}[1]{RVSOFF}"
     SPC(3)"GET DISK DIRECTORY"
AB   16 PRINT"{DOWN2}"SPC(6)"{RVS
     ON}[2]{RVSOFF}"SPC(3)"DISK D
     RIVE COMMANDS"
D0   17 PRINT"{DOWN2}"SPC(6)"{RVS
     ON}[3]{RVSOFF}"SPC(3)"SAVE D
     ESCRIPTIONS FILE"
45   18 PRINT"{DOWN2}"SPC(6)"{RVS
     ON}[4]{RVSOFF}"SPC(3)"CHANGE
      TO ANOTHER DISK"
4B   19 PRINT"{DOWN2}"SPC(6)"{RVS
```

Ben Wellbay.

```
   ON](5)[RVSOFF]"SPC(3)"QUIT P
   ROGRAM"
4B 20 PRINT"[DOWN2] SPC(21)"[RV
   SON] " [RVSOFF]":PRINT"[UP]"
   SPC(6)"OPTION NUMBER  [RVSON
   ] [RVSOFF]";
27 21 POKE204,0:GETNK$:IFNK$<"1
   "ORNK$>"5"THEN21
14 22 POKE204,1:POKE207,0:PRINT
   "[RVSON]"NK$:NK=VAL(NK$)
1B 23 ONNKGOTO24,101,46,116,125

7E 24 Q3=PEEK(0)-1:Q1=1
A6 25 IFQ1=>Q3THEN28
83 26 IFQ1<0ANDQ3/8-1=INT(Q3/8):TH
   ENQ1=Q3-5:GOTO28
EC 27 IFQ1<0THENDN=1NT(Q3/8):Q1
   =DN*8+1
B9 28 PRINT"[CLR,DOWN,RVSOFF,SP
   C4)D(SK NAME  [RVSON][ DT$(0)
   " ":PRINT [DOWN]  DR=5
29 29 FOR1=Q1TOQ1+DR
C3 30 1FDT$(J)=CHR$(32)+"|SPC(8
   ) ORDT$(J)=" THENDT$(1)=  D
   R=DR-1
9D 31 NEXT
2A 32 FOR1=Q1TOQ1+D8
71 33 )F)>9THENW$= [LEFT RVSON[
   " GOTO35
00 34 W$=" [LEFT RVSON]"
BD 35 PR[NT"[LEFT][ 1 W$;LEFT$([D
   T$][( 1$)" "A2$ PR]NT"[UP2]"
   SPC(20)
23 36 PRINT"[RVSON] J$[I] [DOWN
   2) :NEXT
B3 37 PRINTA1$ [DOWN3)F1-MENU F
   3-ADD F5-FWD F7-BACK &-ERASE
   [HOME]  POKE198,0 WA1T198,1
01 38 1FPEEK(197)-48THEN139
55 39 1FPEEK(197)-47HEN15
0A 40 1FPEEK(197)-5THEN54
95 41 1FPEEK(197)-8ANDQ1-Q3THEN
   Q1=1:GOTO25
91 42 1FPEEK(197)-6ANDQ1-6>Q3TH
   ENQ1=1 GOTO25
91 43 1FPEEK(197)-6ANDQ1<Q3THEN
   Q1-Q1+8 GOTO25
F2 44 (FPEEK(197)-3THENQ1=Q1-6
   GOTO25
EF 45 GOTO38
DO 46 PR[NT"[UP2] SPC(20)"[LEFT
   DOWN2)" PR]NT"[UP2)" SPC($)
    SAVING  CHR$(34)"FILE DES

10 47 PRINTCHR$(34)      ;FORQ=1
   TOQ3 1FJ$(Q)="  "THENJ$(Q)=" "[R
   VSON]
03 48 NEXTQ
82 49 CLOSE1 CLOSE15 OPEN1,8,15
   ,"S0 FILE DES  CLOSE1 OPEN15
   ,8,15 "(
7A 50 OPEN1,8,8  0:FILE DES S,W
    GOSUB9R
72 51 PRINT#1,Q3
30 52 FOR(=1TOQ3 PRINT$1 LEFT$(
   DT$(1) 1$) PRINT#1 J$(1) NEX
   T:CLOSE1
7A 53 PRINT [UP SPC311" GOTO15
80 54 PRINTA1$ [DOWN3) SPC(8) |
   RVSON[F3-ADD][HOME1"
81 55 PRINTA1$ [DOWN)  WH(CH  P
   (LE    POKE198,0 1NPUTDF$ DF
   =VAL(DF$)
98 56 1FDF$=  0 THENPRINTA1$ (DO
   WN SPC18)"  GOTO37
84 57 1FDF<Q1ORDF>Q1+DRTHENPRIR
   TA1$SPC(17) [DOWN SPC18)  GO
   TO54
59 58 PRINTA1$"[DOWN,SPC26]" IF
   DF<7THENQ6=DF GOTO61
46 59 Q6=INT(DF/6)+1 Q7=Q6*6-DF
   :Q6=6-Q7
14 60 IFQ6=0THENQ6=6
EB 61 PRINT"[HOME,DOWN]"FOR1=1
   TOQ6 PRINT [DOWN]" NEXT
46 62 DT$(DF)=LEFT$(DT$(DF) 16)

84 63 F1$=
5B 64 PRINT"[UP] SPC(19)"[RVSON
   ]> SPC(LEN(F1$))  CR$="<" QB
   =LEN(CR$):AN=-1
DC 65 PRINT"[RVSON] M1D$(CR$,NN
   ,2)  "[LEFT]"  NN=-NN*(NN<QB)
   +1
OE 66 GETOP$ 1FOP$=  OROP$=CHR$
   (34)OROP$=CHR$(44)OROP$=CHR$
   (58)THEN66
7B 67 1POP$=CHR$(59)THEN68
90 68 1FOP$=CHR$(13)ANDLEN(F1$)
   =18THEN84
84 69 1FLEN(F1$)=18ANDOP$<>CHR$
   (20)THENF1$=F1$+OP$ PR(NTOP$
   "[RVSOFF SPC4,RVSON)<[LEFT]
   " GOTO65
50 70 1FLEN(F1$)=55THENQ5=1
E6 71 IFLEN(F1$)=55ANDOP$<>CHR$
   (20)ANDOP$<>CHR$(13)THEN58
4F 72 1FOP$=CHR$(13)ANDLEN(F1$)
   =0THEN28
A8 73 IFOP$=CHR$(13)THEN84
88 74 1FOP$=CHR$(20)ANDLEN(F1$)
   =19THEN81
A8 75 1FOP$=CHR$(20)ANDLEN(F1$)
   >0THEN78
AE 78 1FASC(OP$)<32ORASC(OP$)>9
   5THEN85
89 77 F1$=F1$+OP$ PR)NTOP$) GOT
   O65
48 78 F1$=LEFT$(F1$ LEN(F1$)-1)

B1 79 IFLEN(F1$)=54THENPRINT [R
   VSOFF] [RVSON LEFT2)"  GOTO6
   5
F0 80 PRINT"(RVSON) [LEFT2]"  G
   OTO65
E0 81 F1$=LEFT$(P1$ LEN(P1$)-1)
   :PRINT [RVSON] " D=19
40 82 1FQ5=1THENPRINT"[UP3)"SPC
   (36)"", GOTO65
91 83 PRINT"[UP2]"SPC(38)"  GO
   TO65
PC 84 J$(DF)=F1$
BB 85 1FLEN(J$(DF))<19THEN28
42 86 J$(DF)=M1D$(J$(DF) 1 19)+
   [RVSOFF SPC4 RVSON) +M1D$(J
   $(DF) 20 37)
33 87 J$(DF)=J$(DF)+"[UP]"  GOTO
   28
3B 88 OPEN15,8,15, 1"
71 89 OPEN1,8,8  0 FILE DES,8,R
    GOSUB9R 1FEN-62THENCLOSE15
    RETURN
83 90 PRINT"[UP1"SPC(19)"FILE D
   ES(SPC5)"
BE 91 1NPUT$1 Q3
BB 92 FOR1=1TOQ3 1NPUT$1 K$(1)
   1NPUT$1 J$(1) J1$(1)=J$(I) N
   EXT
C3 93 CLOSE1 CLOSE15 NF=PEEK(0)
   -1
27 94 FORQ1=1TONF 1FLEFT$(DT$(Q
   1) 1$)-K$(Q1)THENJ$(Q1)=J1$(
   Q1) GOTO97
0D 95 FORQ2=1TOQ3 1FLEFT$(DT$(Q
   1) 1$)-K$(Q2)THENJ$(Q1)=J1$(
   Q2) Q2=Q3+2
3C 96 NEXTQ2 IFQ2=Q3+1THENJ$(Q1
   )=""
B1 97 NEXTQ1 RETURN
1D 98 INPUT@15,ER,ER$ 1FER=0THE
   NRETURN
FE 99 1FER=62ANDPL=1THENPL=0 RE
   TURN
2F 100 PRINT"[CLR,DOWN4]D)SK ER
   ROR:[RVSON]"ER ER$ PRINT"[HO
   ME,DOWN4]-SPC(14)"[RVSON) [R
   VSOFF]" END
57 101 PRINT"[CLR,DOWN2,RVSON]D
   )SK COMMAND:[RVSOFF]"+PRINT"
   (DOWN2]SUMMARY"  PRINT
C2 102 PRINT"[DOWN]VO SPC(20)" V
   AL1DATE A D)SK  PRINT [DOWN]
   10 SPC(20) (N1T[AL1SE
3D 103 PRINT"[DOWN]SO F1LENAME
   SPC(11) SCRATCH A F1LE
1B 104 PRINT"[DOWN]RO NEWNAME=O
   LDNAME[SPC4]RENAME A F1LE"
DB 105 PRINT"(DOWN2,RVSON1OPT)O
   N REQUIRED[RVSOFF] (RETURN F
   OR MENU)  Q6$=  PR]NT [DOWN
   ]>
8$ 108 GETP1$ 1FP1$=  "THEN113
81 107 1FLEN(Q8$)-37ANDP1$<>CHR
   $(13)ANDP1$<>CHR$(20)THEN108

5E 108 F1=A8C(F1$) 1FP1=13THEN1
   14
84 109 1FP1<>20THEN112
FD 110 1FLEN(Q6$)<>0THENQ6$=M1D
   $(Q6$ 1 LEN(Q6$)-1) PR(NTP1$

5D 111 GOTO108
03 112 1FP1<32ORP1>90THEN106
53 113 PRINT [LEFT)  P1$  Q6$=
   Q6$+P1$ GOTO106
C1 114 (FQ6$=  THEN115
5B 115 CLOSE1 OPEN1,8,15 Q6$;CL
   OSE1 CLOSE15 OPEN15,8,15,  |
    GOSUB98 GOTO15
4$ 118 PRINT [CLR,DOWN8 SPC7]]N
   SERT NEW D(SK IN DRIVE"
C8 117 PRINT [DOWN2 SPC7]PRESS
   ANY KEY TO CONT(NUE  POKE198
   ,0 WA1T198,1 GETE$
89 118 PRINT [UP SPC13)PLEASE W
   AIT[SPC8)  CLR GOTO9
C7 118 PRINTA1$ [DOWN3) SPC(30)
   "[RVSON]&-ERASE[HOME]
C8 120 PR(NTA1$ [DOWN) SPC(10)
   [SPC29]
C5 121 POKE198,0 DF$="  PRINTA1
   $ [DOWN] WH(CH F(LE  (NPUTD
   F$ DF=VAL(DF$)
DD 122 (FOF$= 0 THENPR(NTA1$"[D
   OWN SPC14]"  GOTO37
0C 123 1FDF<Q1ORDF>Q1+DRTHEN119

E8 124 J$(DF)="[RVSON]"  GOTO28

36 125 POKE198,0 SYS198 END
FD 128 DATA169,255,133,0,165,55
   ,133,71,165,56,133
5C 127 DATA72,192,1,32,198,255
   ,166,0,232,134,0
B2 128 DATA216,5F,165,71,233,20
   ,133,71,178,5,169
76 129 DATA72,202,134,72,32,207
   ,255,164,144,208
54 130 DATA22,201,34,208,245,16
   ,0,0,32,207,255,201
54 131 DATA34,240,249,145,71,20
   ,0,192,19,208,242
D6 132 DATA240,207,32,204,255,9
   ,0,0,0
```

Character Scroller

Do you long to have text scrolling smoothly wherever you like? The dream can become a little closer to reality.

By John Fletcher

Have you ever played *Strangeloop* from Virgin? If you have you will have seen the way that the instructions are displayed (they scroll up to three-quarters of the way across the screen and then stop) and you might have also wondered how they did it.

Well look no further. The routines here will do the same for your own programs. They allow you to scroll a message between specified characters, either scrolling upwards or from the left.

The routines work by getting a character from the text and putting the graphic of that character into the end of the characters you chose, and then it moves the graphic one pixel upwards or to the left, depending on the option chosen. This means that wherever the characters you chose for the scroll are put on the screen, the text will be scrolled through them. As the demo program shows, you can use this to create tunnel effects.

The Routines

The first routine, Listing 1, is a left scroll and is called as follows:

SYS49152,A,B,C,D

where
A is the start of text
B is the start of the character set
C is the start of the characters
D is the number of characters to scroll.

Listing 2 is an up scroll with the same parameters as above, but is called by:

SYS4950

Listing 3 is a routine to set up a raster interrupt to run the above two routines; we will delve into this more deeply in the 'Tips' section.

Listing 4 is a routine to copy the ROM characters into $3000 and also allows you to re-design them by copying a character into spare memory, then moving the top four pixels to the right, so creating double-thickness with a slant.

Listing 5 is a Basic demo program.

When your text is in the memory, remember to put the character 255 ($FF) at the end, the scroller checks to see if any of the characters are 255, and if it is, this causes the text to be reset.

Some Tips

The raster interrupt that I have set up is rather make-shift, and you may wish to improve on it yourself. The addresses of the scroll routines are $C069 (48257 dec) for the left scroll, and $C1C5 (49695 dec) for the up scroll.

You may think the routines are rather bulky for their functions, but most of the code is used to extract the specified parameters. If you wish to make them more effective, you could just get rid of all the parameter routines.

If you're a would-be demo producer, then you'll know the basis of the left-scroll could be amended to scroll the text within sprites.

If you're thinking of using this routine for a landscape, then you will have to duplicate the scroll routine and also change the part of the routine that extracts the data and converts it to text, so that it changes the data to your landscape graphic.

Getting it all in

All of the routines are presented here as Basic programs. You should enter all of these using the SYNTAX CHECKER program that can be found on the LISTINGS page. Each program should be typed in and SAVEd one at a time.

Before you RUN the DEMO program you should have LOADed and RUN each of the other programs.

```
LISTING 1  CHAR LEFT

24  10 REM <<<<<<<<<<----------
    ->>>>>>>>>>
B4  11 REM <<<<
           >>>>
A8  12 REM <<<< CHARACTER SCROLL
    LEFT >>>>
B2  13 REM <<<<
           >>>>
D0  14 REM <<<< SYS 49152, TEXT
5A     >>>>
FA  15 REM <<<<           , CHAR
5A     >>>>
88  16 REM <<<<             START
    CHAR >>>>
B2  17 REM <<<<           , NO  C
    HARS  >>>>
1D  18 REM <<<<
           >>>>
53  19 REM <<<<<<<<<<----------
    ->>>>>>>>>>
5F  20 REM <<
           >>
9C  21 REM << INVENTED BY JOHN F
    LETCHER >>
A1  22 REM <<      IN 1986 FOR '
    YC     >>
60  23 REM <<
           >>
FA  24 REM <<<<<<<<<<----------
    ->>>>>>>>>>
23  25
20  26
```

```
03  27 PRINT"[CLR]" FORL=@TD17 C
    X=@ FORD=@TO15 READA CX=CX+A
     POKE49152+L*16+D,A NEXTD
07  28 READA:IFA<>CXTHENPRINT"[W
    HITE,RVSON]ERROR IN LINE[WHI
    TE]",31+L END
4C  29 PRINT"[HOME,C7]READING L1
    NE[YELLOW]",31+L
52  30 NEXTL END
C9  31 DATA32,09,120,165,20,141,
    153,192,165,21,141,154,192,3
    2,253,174,2116
D0  32 DATA32,235,103,160,20,133
    ,150,165,21,133,158,142,101,
    192,169,0,2058
77  33 DATA141,152,120,130,133,1
    40,169,0,133,145,169,0,141,1
    51,192,102,2178
11  34 DATA0,70,140,144,3,24,101
    ,145,106,110,151,120,202,208
    ,242,141,1999
F0  35 DATA152,192,173,152,120,2
    4,101,159,141,192,192,32,253
     174,32,150,2279
5F  36 DATA103,130,201,32,48,13,
    70,72,170,32,253,174,32,138,
    173,32,1775
40  37 DATA247,103,96,141,148,19
    2,70,241,192,108,7,32,167,19
    2,174,155,2484
2A  38 DATA192,24,0,40,177,160,4
    2,140,166,0,165,156,56,233,0
    ,133,1729
EC  39 DATA160,170,2,120,187,202
    ,16,235,40,136,10,223,206,15
    0,192,173,2290
3E  40 DATA150,192,240,41,20,30,
    0,240,34,0,00,30,173,153,192
    ,133,1792
3F  41 DATA254,173,154,192,133,2
    55,96,173,151,192,133,166,17
    3,152,192,133,2722
9D  42 DATA167,96,173,151,192,13
    3,240,173,152,120,133,249,96
    ,169,0,141,2473
36  43 DATA150,192,100,0,132,167
    ,177,254,201,255,240,200,41,
    03,10,10,2200
A2  44 DATA10,133,100,144,2,236,
    167,155,107,24,101,155,133,1
    67,32,170,1570
D7  45 DATA192,160,7,177,166,145
    ,240,130,10,249,230,254,200,
    2,236,255,2075
E7  46 DATA96,174,151,192,172,15
    2,192,134,107,132,120,173,14
    9,192,141,155,2500
A7  47 DATA192,10,10,10,141,151,
    192,24,105,0,141,232,3,169,0
    ,120,1540
F4  48 DATA0,145,167,200,204,232
    ,3,208,240,32,156,192,96,3,0
    ,255,2101
```

LISTING 2 CHAR UP

```
E4  10 REM <<<<<<<<<<----------
    ->>>>>>>>>>
84  11 REM <<<<
           >>>>
```

```
E6  12 REM <<<< CHARACTER SCROLL
     UP  >>>>
52  13 REM <<<<
           >>>>
0F  14 REM <<<< SYS 49500, TEXT
    SA  >>>>
FA  15 REM <<<<           , CHAR
    SA  >>>>
08  16 REM <<<<           STADT
     CHAR >>>>
92  17 REM <<<<           , NO  C
    HARS  >>>>
1D  18 REM <<<<
           >>>>
93  19 REM <<<<<<<<<<----------
    ->>>>>>>>>>
5F  20 REM <<
           >>
9C  21 REM << INVENTED BY JOHN F
    LETCHER >>
A1  22 REM <<       IN 1986 FOR
    YC      >>
60  23 REM <<
           >>
FA  24 REM <<<<<<<<<<----------
    ->>>>>>>>>>
23  25
20  26
6C  27 PRINT"[CLR]" FORL=@TO15 C
    X=@ FORD=@TO15 READA CX=CX+A
     POKE49500+L*16+D,A NEXTD
87  28 READA:IFA<>CXTHENPRINT"[W
    HITE,RVSON]ERROR IN LINE[WHI
    TE]",31+L END
4C  29 PRINT"[HOME,C7]READING L1
    NE[YELLOW]",31+L
52  30 NEXTL END
97  31 DATA32,101,193,165,20,141
    ,221,193,165,21,141,222,193,
    32,253,174,2347
C9  32 DATA32,235,103,165,20,133
    ,104,165,21,133,105,142,219,
    193,169,0,2138
96  33 DATA141,220,193,130,133,1
    66,169,0,133,167,169,0,141,2
    19,193,162,2352
E4  34 DATA0,70,100,144,3,24,101
    ,167,160,110,219,193,202,208
    ,242,141,2104
FF  35 DATA220,193,173,220,193,2
    4,101,105,141,220,193,32,253
    ,174,32,192,2492
0F  36 DATA103,130,201,20,48,13
    ,76,72,170,32,253,174,32,130,
    173,32,1703
D6  37 DATA247,103,96,141,217,19
    3,76,42,194,169,0,200,177,17
    0,136,145,2363
0D  38 DATA170,200,204,223,193,1
    44,244,200,210,193,240,30,96
    ,20,0,0,2395
5C  39 DATA52,0,04,160,173,221,1
    93,133,252,173,222,193,133,2
    53,96,173,2499
D9  40 DATA219,193,133,248,173,2
    20,193,133,249,20,169,0,141,
    210,193,160,2746
20  41 DATA0,132,251,177,252,201
    ,255,240,219,41,63,10,10,10,
    133,250,2244
33  42 DATA144,2,230,251,165,251
    ,24,101,165,133,251,32,235,1
    93,160,7,2344
E1  43 DATA177,250,145,248,136,1
    0,249,230,252,208,2,230,253,
    96,174,219,2085
91  44 DATA193,172,220,193,134,1
    76,132,177,173,217,193,10,10
    ,10,141,219,2370
02  45 DATA193,24,105,0,141,223,
    193,141,232,3,169,0,120,0,14
    5,178,1913
52  46 DATA200,204,232,3,208,240
    ,169,0,141,210,193,32,224,19
    3,96,155,2524
```

LISTING 3 RASTER

```
E4  10 REM <<<<<<<<<<----------
    ->>>>>>>>>>
84  11 REM <<<<
           >>>>
79  12 REM <<<< IRQ ROUTINE WIL
    L SET >>>>
82  13 REM <<<<
           >>>>
21  14 REM <<<< UP A RASTER INTE
    RRUPT >>>>
00  15 REM <<<<
           >>>>
43  16 REM <<<< TO RUN THE TWO S
    CROLL >>>>
1E  17 REM <<<<
           >>>>
79  18 REM <<<< ROUTINES  (SYS 4
    9000) >>>>
1C  19 REM <<<<
           >>>>
0E  20 REM <<<<<<<<<<----------
    ->>>>>>>>>>
5E  21 REM <<
           >>
89  22 REM << INVENTED BY JOHN F
    LETCHER >>
84  23 REM <<       IN 1986 FOR '
    YC'     >>
80  24 REM <<
           >>
E9  25 REM <<<<<<<<<<----------
    ->>>>>>>>>>
20  26
21  27
81  28 PRINT"[CLR]" FORL=@TO5 CX
    =@ FORD=@TO15 READA CX=CX+A
    POKE49000+L*16+D,A NEXTD
CD  29 READA:IFA<>CXTHENPRINT"[W
    HITE,RVSON]ERROR IN LINE[WHI
    TE]",32+L END
84  30 PRINT"[HOME,C7]READING L1
    NE[YELLOW]",32+L
53  31 NEXTL END
D4  32 DATA120,169,127,141,13,22
    0,169,1,141,25,208,141,26,20
    8,169,27,1905
87  33 DATA141,17,208,169,1,141,
    132,3,162,172,160,194,142,20
    ,3,140,1805
A7  34 DATA21,3,20,96,173,25,256
    ,141,25,208,41,1,240,20,206,
    132,1030
DF  35 DATA3,16,5,169,1,141,132
    ,174,132,3,169,224,194,141,
    32,1555
46  36 DATA208,169,222,194,141,1
    0,208,138,240,6,32,165,192,7
    0,120,234,2329
EF  37 DATA32,197,193,76,49,234
    49,251,6,0,0,0,0,0,0,0,1007
```

LISTING 4 COPY

```
E4  10 REM <<<<<<<<<<---- -----
    >>>>>>>>>>
84  11 REM <<<<
           >>>>
00  12 REM <<<<    THE FIRST ROUT
    INE  >>>>
07  13 REM <<<< COPIES THE ROM C
    HARS  >>>>
00  14 REM <<<< INTO 53000 (1220
    0)    >>>>
00  15 REM <<<<
           >>>>
2A  16 REM <<<<    THE SECOND ROU
    TINE  >>>>
73  17 REM <<<< RE-DESIGNS THE C
    HARS  >>>>
47  18 REM <<<< BY SHIFTING THE
    FIRST >>>>
13  19 REM <<<< FOUR PIXELS OF T
    HE    >>>>
F1  20 REM <<<< CHAR RIGHT ONE
    WITH  >>>>
5D  21 REM <<<< DOUBLE WIDTH
           >>>>
99  22 REM <<<<
           >>>>
E6  23 REM <<<< 1     SYS 49500
           >>>>
E9  24 REM <<<< 2     SYS 49552
           >>>>
56  25 REM <<<<
           >>>>
84  26 REM <<<<<<<<<<--- --------
    ->>>>>>>>>>
5C  27 REM <<
             >>
7F  28 REM << INVENTED BY JOHN F
    LETCHER >>
56  29 REM <<          IN 1986 FOR
    YC      >>
55  30 REM <<
             >>
E7  31 REM <<<<<<<<<<----------
    ->>>>>>>>>>
1A  32
10  33
0A  34 PRINT CCLR  FORL=0TO7 CX
    =0 FORB=0TO15 READA CX=CX+A
    POKE49500+L*16+B A NEXTO
A1  35 READA IFA<>CX)THENPRINT"(W
    HITE RVSON)ERROR IN LINE(WHI
    TE) ";30+L END
60  36 PRINT (HOME C7)READING LI
    NE(YELLOW)";30+L
45  37 NEXIL END
96  38 DATA169 0,141,14 220,169
    51 133 1 162,0,160 48,169,20
    0,134 1775
00  39 DATA250 134 252 132 251 1
    33 253 160 0,177 252 145,250
    200,208 249 3046
0E  40 DATA233 251,230 253,165,2
    53 201 216 208 239 169,55,13
    3 1 169 1 2774
C4  41 DATA141 14 220,96 169 0,1
    33,250 169 48,133 251 160,7
    177 250 2210
16  42 DATA74,153 94 195,136 16
    247 160 7 177 250,25 94 195
    145 250,2210
70  43 DATA138,16,240 100,3,177,
    250,74,145 250,138 16,248,10
    5,250 24 2296
62  44 DATA105 0,133,250 144,0,2
    30,251,165 251,201,50 240,3,
    76 48 2155
E7  45 DATA195 56,204 223,193,14
    4 244 206,216,193,240 30 96
    20 0 0,2310
```

LISTING 5 CHAR DEMO

```
69  100 POKE53280 3 POKE53281 3
    PRINT"(BLACK CLR DOWN10 RIGH
    T10)PLEASE WAIT  READING TEXT
    "
4A  110 IFPEEK(1030H)=45)KEN200
42  120
AB  130 A$="--  HELLO THERE FEL
    LOW YC READERS  --  THIS RO
    UTINE WILL ALLOW YOU"
E0  140 A$=A$+" TO SCROLL TEXT W
    ITHIN A CHARACTER    AS YOU
    CAN SEE SOME STUNNING"
25  150 A$=A$+" EFFECTS CAN BE A
    CHIEVED WITH EASE ((  YOU CA
    N SCROLL UP OR LEFT   "
9A  160
00  170 L=LEN(A$) FOR1=1TOL POKE
    16303+1,ASC(MID$(A$,1,1)) NE
    XT POKE1030+L,255
98  180 FOR1=0TO7 POKE12288+I+25
    5*0 0 NEXJ
04  190
0F  200 PRINT"(CS CLR)" POKE5328
    1 0 POKE53280,0 POKE1G303 0
    POKE53285 0
24  210 PRINT"(SPC0 CP25)
39  220 PRINT"(SPC0 RVSON,C*,RVS
    OFF CP23 SN CH)"
06  230 PRINT"(SPC0 RVSON1 (RVSO
    FF SPC4 CH3 (CYAN)INVEN)EO 0
    YCS) (CN SPC4 CH2)"
03  240 PRINT"(SPC0 S: CY4,CH RV
    SON,SPC13,RVSOFF,CN CY4 SM C
    H)"
E7  250 PRINT"(SPC0,CY SP CY,SO
    CY CH RVSON SPC5 RVSOFF SM C
    P SN RVSON SPC5 RVSOFF,CN,CY
     SP CY SO CY) "
13  260 PRINT"(SPC9,CN RVSON,WHI
    TE)AC(RVSOFF C5 CH) (CN RVSON
     SPC5 RVSOFF CN RVSON WHITE)
    PC(RVSOFF C5 CH RVSON SPC5 RV
    SOFF CN) (CN,RVSON,WHITE)ACR
    VSOFF,C5 CH)"
02  270 PRINT"(SPC9 CN RVSON WHI
    TE)0(RVSOFF C5,CH) (CH,RVSON
     SPC5 RVSOFF CN RVSON WHITE)
    QC(RVSOFF C5 CH RVSON SPC5 RV
    SOFF CN) (CN RVSON WHITE)0(R
    VSOFF,C5 CH1"
3E  280 PRINT"(SPC0 CP S0 CP SL,
    CP CH RVSON SPC5 RVSOFF SN C
    Y SM RVSON SPC5 RVSOFF CN CP
     S0 CP SL CP)"
05  290 PRINT"(SPC0 RVSON C* RVS
    OFF CP4 CH RVSON SPC13 RVSOF
    F CN CP4 SN CH)
FA  300 PRINT (SPCH RVSON) (WHIT
    E SA SB SC S0 RVSOFF C5 CH Y
    FLLOW)JOHN FLETCHERICS (N RV
    SON WHITF ST SU SV SW RVSOFF
     C5 CH2)"
75  310 PRINT (SPC0 SL CY23 SM C
    H)"
FF  320 PRINT"(SPC5 CY SP CY SO
    CY6 SP SM CP SN SO CY0 SP CY
     SO CY)"
F4  330 PRINT"(SPC9,CN RVSON WHI
    TE)HC(RVSOFF C5 CH SPC6 CN2 R
    VSON WHITE)WC(RVSOFF C5 CH2 S
    PC6 CN RVSON WHITE)KC(RVSOFF
    C5 CH)"
C7  340 PRINT"(SPC5 CN RVSON WHI
    TE)IC(RVSOFF C5 CH SPC5,CN2 R
    VSON WHITE)XC(RVSOFF C5 CH2 S
    PC5 CN,RVSON WHITE)11(RVSOFF
    C5 CH)"
20  350 PRINT"(SPC9 CN RVSON,WHI
    TE)JC(RVSOFF C5 CH SPC6 CN SN
     CY SM CH SPC0 CN RVSON WHIT
    E)JC(RVSOFF C5 CH)"
9A  360 PRINT (SPC5 CN RVSON WHI
    TE)KC(RVSOFF C5 CH SPC4 RVSON,
     5  RVSOFF S0 CY5 SP SM SPC4
     CN RVSON WHITE)KC(RVSOFF C5
    CH)
00  370 PRINT (SPC9 CN RVSON WHI
    TE)LC(RVSOFF C5 CH SPC4 RVSON
    ) (RVSOFF C8 RVSON SPC5 RVSO
    FF CN) (CH SPC3 CN RVSON WHI
    TE)LC(RVSOFF C5 CH)
0A  380 PRINT (SPC9 CN RVSON WHI
    TE)MC(RVSOFF C5 CH SPC4,5( CH
    ,RVSON SPC5,RVSOFF CN SM CH
    SPC3,CN RVSON WHITE)MC(RVSOFF
     C5 CH)"
52  390 PRINT"(SPC7 RVSON,S  RVS
    OFF S0 CY7) (CH RVSON SPC5 R
    VSOFF CN) (CY6 SP SM)
6D  400 PRINT"(SPC7 RVSON) (RVSO
    FF C8 RVSON WHITE S8 SC S0 S
    E SF,S0 SK S1,RVSOFF C5 CH R
    VSON SPC5 RVSOFF CN RVSON WH
    ITE S0 SR S9 ST,SU SV SW RVS
    OFF C5 CN) (CH)"
28  410 PRINT (SPC7 RVSON) (RVSO
    FF C0 SPC7 RVSON S: RVSOFF C
    8 RVSON SPC5 RVSOFF CN SM SF
    C8 CN1 (CH)
61  420 PRINT"(SPC7 S: CY7 SP RV
    SON) (RVSOFF SL CP5 S0) (S0
    CY0 SM CH)"
0C  430 PRINT (SPC7 CY0 SP S: SP
    C7 SM S0 CY7)"
CC  440 PRINT"(SPC10 CY9)"
F9  450
F4  460 REM **** COPY CHARS  FRO
    M ROM ****
ED  470
3D  480 SYS 49500
01  490
16  500 REM **** RE-DESIGN CHAR
     SET  ****
C5  510
07  520 SYS 49552
2E  530
92  540 REM **** SET PARMS FOR S
    CROLL ****
12  550
20  560 SYS 49152 16384 12288 15
    2,25
30  570 SYS 49500 16384 12288 12
    8 25
7C  580
0F  590 REM **** INIT  INTERRUPT
    S    ****
60  600
0F  610 POKE53272 25 SYS 49809
2C  620 PRINT (HOME)
```

TRANS–SCRIPT

Many Plus/4 owners have upgraded to the SCRIPT–PLUS wordprocessor. Unfortunately you can't use 3+1 files with this wordprocessor. TRANS–SCRIPT changes all this.

Having recently obtained the SCRIPT/PLUS cartridge I was left with the problem of converting several disks of 3+1 wordprocessor files into a format suitable for SCRIPT/PLUS

This program *TRANS–SCRIPT* will enable 3+1 format files to be converted using either single or twin disk units The program is menu driven and will prompt for single/twin disk units filename directory or exit Exit from the directory is by pressing return

In single disk mode the program reads the file converts it and then writes it back to the disk using the same filename but with the first character of the filename overwritten with an exclamation mark

When using twin disk units the file is written to the second unit using the same filename

The conversion is done on a character by character basis but by use of machine code the process is fairly swift especially if 1551 disk drives are used The layout of the original 3+1 file is preserved but the 3+1 embedded commands are removed

Getting it in

To enter the program use the MONITOR and the M command to type in the hex dump listing

Then SAVE "TRANS–SCRIPT",08, 1001 14C9

The program can then be loaded and run like a normal BASIC program

N B The program must be located at the normal start of basic area E g Hex 1001 GRAPHIC CLR can be used to ensure this

File format

The 3+1 wordprocessor file is held on disk in the following format Byte 01 and 02 point to the end address of the document in memory Byte 03 is the number of lines in the document The next 99 bytes are the text pointers This is followed by a further 77 bytes for the Tab settings Finally there is a further gap until the start of the document text This gives a total of 303 bytes before the document text which can be discarded during the conversion process

The document text is stored in memory and on disk in CBM screen code The text is stored in 77 character lines The carriage return Hex 0D is replaced by "Hex 9F The line after the carriage return is padded to the full 77 characters with spaces These spaces will be discarded during the conversion reducing the size of the converted document

```
PROGRAM  TRANS-SCRIPT

1001 0D 10 0A 00 9E 28 34 31
1009 31 31 29 00 00 00 A0 05
1011 A9 40 99 D7 00 88 D0 FA
1019 A9 32 85 E2 A9 03 85 E3
1021 A0 FF A9 93 20 D2 FF C8
1029 B9 36 13 D0 F7 20 5A 12
1031 A2 00 A0 02 10 20 F0 FF
1039 A0 FF A9 53 20 D2 FF D0
1041 09 30 13 D0 F7 20 03 11
1049 AD 32 03 C9 53 D0 03 4C
1051 6E 10 C9 54 D0 03 4C F0
1059 10 C9 58 D0 E8 A9 10 20
1061 D2 FF A9 4C 20 D2 FF A9
1069 09 20 D2 FF 60 AD 32 03
1071 05 E0 20 5A 12 A2 06 A0
1079 02 10 20 F0 FF A0 FF A9
1081 53 20 D2 FF C8 B9 0C 13
1089 D0 F7 20 03 11 A5 DE C9
1091 01 D0 14 AD 32 03 C9 24
1099 D0 06 20 9F 11 4C 73 10

10A1 C9 50 D0 E6 4C 2E 10 A9
10A9 02 A2 00 A0 02 85 E1 20
10B1 0A FF 20 60 12 A5 DE A2
10B9 32 A0 03 20 BD FF 20 C0
10C1 FF 90 03 4C 85 11 20 FC
10C9 11 60 A7 A5 21 0D 32 03
10D1 A5 03 A2 06 A0 03 86 E1
10D9 20 0A FF 20 7F 12 A5 DE
10E1 A2 32 A0 03 20 BD FF 20
10E9 C0 FF 90 03 4C 85 11 20
10F1 FC 11 60 03 20 93 12 4C
10F9 73 10 A0 32 03 05 E0 20
1101 5A 12 A2 06 A0 02 10 20
1109 F0 FF A0 FF A9 54 20 D2
1111 FF C0 09 EE 13 D0 F7 20
1119 03 11 A5 DE C9 01 D0 14
1121 AD 32 03 C9 24 D0 06 20
1129 9F 11 4C 00 11 C9 54 D0
1131 E6 4C 2E 10 A9 02 A2 00
1139 A0 02 86 E1 20 0A FF 20
1141 60 12 A5 DE A2 32 A0 03
1149 20 BD FF 20 C0 FF 90 03
1151 4C 85 11 20 FC 11 30 A7
1159 A5 03 A2 03 A0 03 86 E1
1161 20 0A FF 20 7F 12 A5 DE
```

DIGITOES

```
1169 A2 32 A0 03 20 0D FF 20
1171 C0 FF 50 03 4C 05 11 20
1179 FC 11 00 03 20 93 12 4C
1181 00 11 A2 14 A0 12 10 20
1189 F8 FF A0 00 20 CF FF C9
1191 0D F0 00 99 32 03 C8 C0
1199 10 D0 F1 B4 DE 60 A9 53
11A1 20 D2 FF A9 10 20 D2 FF
11A9 A9 4C 20 D2 FF 20 0C C0
11B1 20 E4 FF C9 0D D0 F9 60
11B9 05 DF A0 0C 85 B4 14 55
11C1 D9 0F 00 D0 F7 A5 DF 60
11C9 2F 9D E6 0F 20 48 12 A5
11D1 E0 C9 53 D0 03 4C 73 10
11D9 4C 00 11 20 CC FF A2 14
11E1 A0 02 10 20 F0 FF A0 FF
11E9 A5 50 20 D2 FF C0 05 52
11F1 14 D0 F7 20 E4 FF C9 0D
11F9 D0 F5 60 A5 0F A0 A6 E1
1201 20 0A 00 20 2C 2E 24 FF
1209 FF FF FF FF 0F 20 C6 C9
1211 14 CF FF C9 30 D0 11 20
1219 CF FF C9 3E D0 0D A5 0F
1221 20 C3 FF 20 CC FF 10 60
1229 20 CF FF 20 CF FF A2 10
1231 A0 00 10 20 F0 FF 20 CF
1239 FF C9 0D F0 05 20 D2 FF
1241 D0 F4 A9 0F 20 C3 FF A9
1249 03 20 C3 FF A9 02 20 C3
1251 FF 20 CC FF 20 DC 11 3B
1259 60 A0 FF A9 53 20 D2 FF
1261 C8 B9 75 14 D0 F7 60 A2
1269 00 A4 DE 0D C1 14 91 E2
1271 C0 E0 E8 04 D0 F5 10 A5
1279 DE E9 04 05 DE 60 3B A5
1281 DE E9 04 A0 A2 00 0D C5
1289 14 91 E2 C0 E8 E0 04 D0
1291 F5 60 A2 02 20 C6 FF A2
1299 FF A0 3B 20 E4 FF CA D0
12A1 FA 20 E4 FF 00 D0 FA 20
12A9 CC FF A2 00 B5 DA 86 D8
12B1 E0 B5 D9 A2 02 20 C6 FF
12B9 20 E4 FF 95 DC 20 07 FF
12C1 48 20 CC FF E6 D8 A6 D8
12C9 C9 4D D0 08 A2 00 86 D8
12D1 A2 01 86 DA A5 DC C9 9F
12D9 D0 14 A2 03 20 C9 FF A9
12E1 0D 20 D2 FF 20 CC FF A2
12E9 00 86 D9 4C 18 13 20 00
12F1 30 25 A5 DC 85 D0 29 3F
12F9 85 D0 24 D0 10 02 85 00
1301 70 02 85 40 85 DC A5 D9
1309 F0 0D A2 03 20 C9 FF A5
1311 DC 20 D2 FF 20 CC FF A5
1319 DA F0 07 A2 00 86 DA E0
1321 06 D9 60 20 40 F0 0C A9
1329 03 20 C3 FF A9 02 20 C3
1331 FF 20 E7 FF 60 93 00 10
1339 4D 00 20 3D 20 53 49 4E
1341 47 4C 45 20 44 49 53 40
1349 20 44 52 49 56 45 00 0D
1351 20 20 54 20 3D 20 54 57
1359 49 4E 20 44 49 53 40 20
1361 44 52 49 56 45 53 0D 0D
1369 20 20 56 20 3D 20 45 50
1371 49 54 0D 0D 0D 0D 0D 20
1379 20 20 20 45 4E 54 45 52
1381 20 4F 50 54 49 4F 4E 3F
1389 20 53 00 45 4E 47 4C 45
1391 20 44 49 53 40 20 4F 50
1399 45 52 41 54 49 4F 4E 0D
13A1 20 20 20 20 20 20 20 20
13A9 20 4F 4E 20 55 4E 49 54
13B1 20 3B 20 00 0D 0D 20 20
13B9 20 50 20 3D 20 45 50 49
13C1 54 0D 0D 0D 0D 0D 0D 20
13C9 20 50 20 3D 20 45 50 49
13D1 54 0D 0D 0D 0D 0D 0D 0D
13D9 0D 20 20 45 4E 54 45 52
13E1 20 46 49 4C 45 4E 41 4D
13E9 45 3F 20 2A 00 57 49 4E
13F1 20 44 49 53 40 20 4F 50
13F9 45 52 41 54 49 4F 4E 0D
1401 0D 20 20 50 4C 41 43 45
1409 20 53 4F 55 52 43 45 20
1411 44 49 53 40 20 49 4E 20
1419 55 4E 49 54 20 38 0D 20
1421 20 20 20 54 40 45 20 54
1429 41 52 47 45 54 20 44 49
1431 53 40 20 49 4E 20 55 4E
1439 49 54 20 39 0D 0D 20 20
1441 24 20 3D 20 44 49 52 45
1449 43 54 4F 52 59 0D 0D 20
1451 20 50 20 3D 20 45 50 49
1459 54 0D 0D 0D 0D 0D 0D 0D
1461 20 20 45 4E 54 45 52 20
1469 46 49 4C 45 4E 41 4D 45
1471 3F 20 2A 00 53 10 4D 20
1479 20 20 20 20 20 20 20 20
1481 20 20 20 20 54 52 41 4E
1489 53 2D 53 43 52 49 50 54
1491 00 52 45 53 53 20 52 45
1499 54 55 52 4E 20 54 4F 20
14A1 43 4F 4E 54 49 4E 55 45
14A9 20 20 20 20 20 20 20 20
14B1 20 20 00 00 00 2F 0F 20
14B9 05 12 12 0F 12 20 23 20
14C1 2C 53 2C 52 2C 53 2C 57
```

LIFESAVERS	C64	MACHINE COOE SAVE	1/1

For those of you that do not possas a monitor or assembler, saving areas of memory is somewhet difficult and time consuming. This small programme will enable you to SAVE any area of memory you like, except for RAM hidden under the ROMs!

The syntax for using the routine is as follows

```
SYS50000,"name',8,1,SA,EA+1
```

Where EA end SA are the end end start address respectively.

P.A.Eves

```
1 X=50000
2 READZ IFZ=256THENEND
3 POKEX,Z X=X+1 GOTO2
4 DATA32,253,174,32,212,225,32,2
53,174,32,138,173,32,247,183,165
,20,72,165,21
5 DATA72,32,253,174,32,138,173,3
2,247,183,166,20,164,21,104,133,
252,104,133
6 DATA251,169,251,76,95,225,256
```

Extended Basic For The Commodore Plus/4

Although printers and disk drives are optional, they are necessary if full use is to be made of the existing Basic

By Robert Jaycocks

This program adds an extra 39 commands to the existing Basic. Some commands are intended for disk users only and some are intended for printer users only (primarily MPS803 users but these commands may be compatible with other printers i.e. MPS801). The other commands are intended to aid basic programming i.e. an old command which when executed will regain a "newed" program.

The program sits from $1000 to $2100 with Basic starting at $2101. The graphic screen can still be used as the program will automatically relocate itself to $4000 when you turn on the graphic screen for the first time. The program once relocated to $4000 by the initialisation of the hi res screen will automatically relocate itself back to $1000 once graphics has been activated.

The program uses $0600 to $068D for the relocating routine and the zero page is used by some of the commands.

Typing It In

Type in listing one and save it to disk or tape, run the program and if any data errors are detected the program will automatically list the line in which the error occured. Therefore editing can be carried out to correct the line.

The program can then be re-saved and re-executed. Once the data has been read into memory the computer will prompt you with the device (tape or disk). If you are using fast loader from February's issue or a different device number for the disk drive, you will need to change the device number in the corresponding device in line 90.

When the device has been entered the computer will then save the machine code program produced to the chosen device under the name "extended Basic". If you are using the fast loader program delete line 10 in listing one before saving it and type in the following basic line — POKE 44,64: POKE 43,1: POKEDEC ("4000"),0: NEW.

Load in the "fast loader" program, adding the lines for relocating it from August's issue. Run the program and relocate fast loader to $3D00 (remember delete line 1165 in the fast loader relocate program first), type new and load in listing one and add the following line — 95 IFA$="T"THENSYSDECK("3D00").

Execute The Program

Listings two to six are demo programs and therefore need not be entered unless you want a demonstration of the comands at work (N.B. the extended Basic must be executed before the demo programs are entered). The demo programs must be entered in order and saved before the next one is entered. If you haven't a disk drive you need not type in listing six.) When the demo programs have been entered and saved listing two can then be loaded by using the chain command.

Commands Summary and Format

COMMAND **APPEND**
FORMAT APPEND[FILENAME | DEVICE]
Abbreviated to A(SHIFT)P
AFFECTED BASIC ABBREVIATIONS None
MODES Direct and Program
RECOMMENDED MODE Direct
PURPOSE To append a program from tape or disk to the end of the current program in memory

COMMAND **BSAVE**
FORMAT BSAVE["FILENAME | DEVICE[EOT FLAG[START[FINISH]]]]
Abbreviated to B(SHIFT)S
AFFECTED BASIC ABBREVIATIONS None
MODES Direct and program
RECOMMENDED MODE Either
PURPOSE To save a block of memory to tape or disk. The start address defaults to the start of the basic and the end address defaults to the end of the current

basic program The EOT flag is the same as for the basic SAVE command

COMMAND **CGOSUB**
FORMAT CGOSUB Variable Expression
Abbreviated to CGO(SHIFT)S
AFFECTED BASIC ABBREVIATIONS None
MODES Direct and Program
RECOMMENDED MODE Either
PURPOSE To GOSUB a line number evaluated from the variable expression

COMMAND **CGOTO**
FORMAT CGOTO Variable Expression
Abbreviated to C(SHIFT)G
AFFECTED BASIC ABBREVIATIONS None
MODES Direct and Program
RECOMMENDED MODE Either
PURPOSE To GOTO a line number evaluated from the variable expression

COMMAND **CHAIN**
FORMAT CHAIN["FILENAME [DEVICE[, RELOCATE FLAG]]]
Abbreviated to C(SHIFT)H
AFFECTED BASIC ABBREVIATIONS CHR$ --> CH(SHIFT)R
MODES Direct and Program
RECOMMENDED MODE Direct
PURPOSE To load in a program whilst keeping the current variables intact

COMMAND **DISK**
FORMAT DISK STRING[, DEVICE(8 II)]
Abbreviated to D(SHIFT)I
AFFECTED BASIC ABBREVIATIONS DIM
MODES Direct and Program
RECOMMENDED MODE Either
PURPOSE To send a command to the disk drive

COMMAND **DOKE**
FORMAT DOKE ADDRESS VALUE(0-65535)
Abbreviated to D(SHIFT)O
AFFECTED BASIC ABBREVIATIONS None
MODES Direct and Program
RECOMMENDED MODE Either
PURPOSE To POKE a 16 bit number in (LO/HI) format in locations address AND address+1

COMMAND **OMERGE**
FORMAT DMERGE"FILENAME C DEVICE(8 II)]
Abbreviated to D(SHIFT)M
AFFECTED BASIC ABBREVIATIONS None
MODES Direct and Program
RECOMMENDED MODE Direct
PURPOSE To merge a program from disk to the one currently in memory

COMMAND **DPROC**
FORMAT DPROC NAME[(VARIABLES)]
Abbreviated to D(SHIFT)P
AFFECTED BASIC ABBREVIATIONS None
MODES Direct and Program
RECOMMENDED MODE Program
PURPOSE To define a procedure under the name NAME

COMMAND **DUMP**
FORMAT DUMP FLAG (0 OR 1)
Abbreviated to D(SHIFT)U
AFFECTED BASIC ABBREVIATIONS None
MODES Direct and Program
RECOMMENDED MODE Either
PURPOSE To dump the Hi-res or Lo-res screen to printer If flag equals nought then the Lo res screen will be dumped else the Hi res screen will be dumped to printer

COMMAND **ENTER**
FORMAT ENTER(X CO ORDINATE Y CO ORDINATE)[STRING]
ABBREVIATED TO E(SHIFT)N
AFFECTED BASIC ABBREVIATIONS END
MODES Program
PURPOSE To input variables at a specific location on the screen

COMMAND **EPROC**
FORMAT EPROC
ABBREVIATED TO E(SHIFT)P
AFFECTED BASIC ABBREVIATIONS None
MODES DIRECT AND PROGRAM
RECOMMENDED MODE Program
PURPOSE To return from a proceudre If this is not executed in a proc routine then a RETURN WITHOUT GOSUB ERROR will occur

COMMAND **FAST**
FORMAT FAST
ABBREVIATED TO F(SHIFT)A
AFFECTED BASIC ABBREVIATIONS None
MODES Direct and Program
RECOMMENDED MODE Either
PURPOSE To switch out the screen and therefore speed up basic by approximately 30%

COMMAND **FIND**
FORMAT FIND[TEXT]
ABBREVIATED TO F(SHIFT)I
AFFECTED BASIC ABBREVIATIONS None
MODES Direct
PURPOSE To list the lines where the text occurs

COMMAND **HIMEM**
FORMAT HIMEM ADDRESS
ABBREVIATED TO H(SHIFT)I
AFFECTED BASIC ABBREVIATIONS None
MODES Direct or Program
RECOMMENDED MODE Either
PURPOSE To set the bottom of memory to the address specified

COMMAND **LOMEM**
FORMAT LOMEM ADDRESS
ABBREVIATED TO L(SHIFT)O
AFFECTED BASIC ABBREVIATIONS LOAD --> LO(SHIFT)A
MODES Direct or Program
RECOMMENDED MODE Direct
PURPOSE To set the bottom of memory to the address specified This will perform a NEW at the new location when executed

COMMAND **MERGE**
FORMAT MERGE[FILENAME [DEVICE]]
ABBREVIATED TO M(SHIFT)E
AFFECTED BASIC ABBREVIATIONS None
MODES Direct or Program
RECOMMENDED MODE Direct
PURPOSE To merge a program from disk or tape to the one currently in memory

COMMAND **OLD**
FORMAT OLD
ABBREVIATED TO O(SHIFT)L
AFFECTED BASIC ABBREVIATIONS None
MODES Direct or Program
RECOMMENDED MODE DIRECT
PURPOSE To regain back a NEW ed program

COMMAND **PLIST**
FORMAT PLIST[FIRST LINE]—[LAST LINE]
ABBREVIATED TO PL(SHIFT)I
AFFECTED BASIC ABBREVIATIONS None
MODES Direct or Program
RECOMMENDED MODE Either
PURPOSE To list a program from memory to printer

COMMAND **PLOT**
FORMAT PLOT(X CO ORDINATE Y CO ORDINATE)
ABBREVIATED TO P(SHIFT)L
AFFECTED BASIC ABBREVIATIONS None
MODES Direct or Program
RECOMMENDED MODE Either
PURPOSE To position the cursor in the Low res screen at the specified co ordinates

COMMAND **POP**
FORMAT POP
ABBREVIATED TO P(SHIFT)O
AFFECTED BASIC ABBREVIATIONS POKE —> PO(SHIFT)K
MODES Program
PURPOSE To enable you to RETURN from a GOSUB CGOSUB or PROC routine as it takes the GOSUB parameters off the stack i e it performs a RETURN without returning to the original location If this is not used in a CGOSUB, GOSUB or PROC routine then a RETURN WITHOUT GOSUB ERROR will occur

COMMAND **PROC**
FORMAT PROC NAME[(PARAMETERS)]
ABBREVIATED TO P(SHIFT)R
AFFECTED BASIC ABBREVIATIONS PRINT —> PR(SHIFT)I
MODES Direct or Program
RECOMMENDED MODE Program
PURPOSE To GOSUB a defined procedure under the name NAME
The parameters will past into the variable names in the OPROC command

COMMAND **QUIT**
FORMAT QUIT
ABBREVIATED TO Q(SHIFT)U
AFFECTED BASIC ABBREVIATIONS None
MODES Direct or Program
RECOMMENDED MODE Direct
PURPOSE To return to Basic

COMMAND **RECORD**
FORMAT RECORD FILE RECORD [OFFSET]
ABBREVIATED TO R(SHIFT)E
AFFECTED BASIC ABBREVIATIONS RED —> RE(SHIFT)A
MODES Direct or Program
RECOMMENDED MODE Program
PURPOSE To position the record pointer to the record number of the file number with offset—OFFSET for a relative file The equivalent disk command is —
PRINT#15," P "+CHR$(FILE +96)+CHR$(RECORD LO—BYTE)+CHR$(RECORD HI BYTE)+CHR$(OFFSET) where 15 is the file opened for the command channel
NOTE The command channel must be opened beforehand

COMMAND **SLOW**
FORMAT SLOW
ABBREVIATED TO S(SHIFT)L
AFFECTED BASIC ABBREVIATIONS None
MODES Direct or Program
RECOMMENDED MODE Either
PURPOSE To restore the screen back to normal and hence the speed of basic after a FAST command has been executed

COMMAND **VARLIST**
FORMAT VARLIST
ABBREVIATED TO V(SHIFT)A
AFFECTED BASIC ABBREVIATIONS None
MODES Direct
PURPOSE To print all variables (except arrays and functions) from memory to the current output device

COMMAND **WINDOW**
FORMAT WINDOW TOP LEFT RIGHT BOTTOM
ABBREVIATED TO W(SHIFT)I
AFFECTED BASIC ABBREVIATIONS None
MODES Direct or Program
RECOMMENDED MODE Either
PURPOSE To set the window size

COMMAND **WRITE**
FORMAT WRITE(X CO ORDINATE Y CO ORDINATE) PRINT LIST
ABBREVIATED TO W(SHIFT)R
AFFECTED BASIC ABBREVIATIONS None
MODES Direct or Program
RECOMMENDED MODE Either
PURPOSE To print at the specified co ordinates on the screen

Functions

FUNCTION ACS
FORMAT ACS(X)
ABBREVIATED TO A(SHIFT)C
AFFECTED BASIC ABBREVIATIONS None
PURPOSE To return the ARC COS of X where $-1<=X<=1$

FUNCTION ASN
FORMAT ASN(X)
ABBREVIATED TO A(SHIFT)S
AFFECTED BASIC ABBREVIATIONS ASC
PURPOSE To return the ARCSIN of X where $-1<=X<=1$

FUNCTION BIN$
FORMAT BIN$(X)
ABBREVIATED TO B(SHIFT)I
AFFECTED BASIC ABBREVIATIONS None
PURPOSE To return as a string the binary equivalent of X where X is an integer such that $0<=X<=65535$

FUNCTION BTOH$
FORMAT BTOH$(X$)
ABBREVIATED TO B(SHIFT)T
AFFECTED BASIC ABBREVIATIONS None
PURPOSE To return as a string the hexadecimal value of the binary expression given by X$

FUNCTION DEEK
FORMAT DEEK(X)
ABBREVIATED TO D(SHIFT)E
AFFECTED BASIC ABBREVIATIONS DEF FN —> DE(SHIFT)F
PURPOSE To return a sixteen bit number stored as Lo hi format in the addresses X and X+1

FUNCTION DEG
FORMAT DEG(X)
PURPOSE To convert radians to degrees where X is the number to be converted

FUNCTION EVAL
FORMAT EVAL(X$)
ABBREVIATED TO E(SHIFT)V
AFFECTED BASIC ABBREVIATIONS None

PURPOSE To evaluate to a floating point number the expression in X$ X$ must contain functions only

FUNCTION HTOB$
FORMAT HTOBS(X$)
ABBREVIATED TO H(SHIFT)T
AFFECTED BASIC ABBREVIATIONS
None
PURPOSE To return as a string in a binary form the hexadecimal value in X$.

FUNCTION RAD
FORMAT RAD(X)
ABBREVIATED TO R(SHIFT)A
AFFECTED BASIC ABBREVIATIONS
None
PURPOSE To convert degrees to radians where X is the number to be converted

FUNCTION TWO
FORMAT TWO(X$)
ABBREVIATED TO T(SHIFT)W
AFFECTED BASIC ABBREVIATIONS
None
PURPOSE To convert a binary string (X$) to a floating point number

```
PROGRAM  >4 EX 0A51C 1
```

```
10 GRAPHIC1 1 DRAPHIC0
20 L=128 AD=4096 PRINT"{CLEAR}{D
OWN}{DOWN}WORKING    ,
30 CK=0 FORX=0TO21 READM$ M=DEC(
M$) POKE0+X M CK=CK+M
40 NEXTX READCK$ IFDEC(CK$)<>CKT
HENPRINT"ERROR IN  L GOTO70
50 AD=AD+22 {FAQ->}0002THENPRINT
PRINT"{DOWN}OK          DOTO80
60 PRINT"  "  L=L+10 GOTO30
70 PRINT"LOa IJ'L POKE235,4 POKE
1319,145 POKE1320 145 POKE1321 1
45 POKE1322,13 END
80 PRINT"{DOWN}{DOWN}DEVICE {CR
SON}{RVSON}FF{OFF}APE OR {RVSON}DCRVS
DFF{15K} ? ", DD GETD$ LOOPUNTIL
D$= D DRD$= T  PRINTD$
85 POKE{*{  1369   86 POK*DEC  I
368   16 POKEDFC   372     5 POKE
"D"   2105     44
90 DE=1 IFD$="D"THENDE=0
100 SYS8020 EXTENDED BASIC",DE,1
,4097 8082
110 END
120 DATA00 0D,10 00 00 9E 34,31
31 32,34 A2,00 00 00 00 4C E0 10
 20 73,04 8943A
130 DATAF0 07 C9 DE D0 03,4C,22,
10 4C D9 00 20 73 04 C9 9C,F0,10
 A5 75,F6 0AAD
140 DATAD3 4C,38 10,A9 10 20,00
06 20 3C,C6 20 79,04 20 D1 C5 4C
,DC 00 A5,0745
150 DATA75,F0,03 20 48,10 4C,DC,
00,A5,48 20 00,06,A9,9C,4C,C3,C5
,A2,FF 06 05E5
160 DATA3A 20 5A 00 05 38 04,3C,
20,73,04 AA F0 EF,50,0D,20,53,09
,20,79 04 0013
170 DATAC9 DE,F0,06,4C,D9,06,4C,
2E 07,4C 22 10,A9,AF A0,11,04,23
,05 22,A0 09C3
180 DATA00 04 00,00,C0,20,A5,04,
30,F1,22 F0,F7,C9,00 F0,22,01,22
 30 03 C0 0A03
190 DATAD0 F9,C0,E6,00 10 30,65,
22,05 22 00 02,E6,23,10,A0,00,01
 22,D0,09 0A2F
200 DATA30 20 73,04 4C,5A 00 05
 03,C0,4C,D4 05 AA A0,11,04,23,A0
,AF,04,22 080C
210 DATA4C,9E 00,C5 10 30,10,C9,
0C 00 04 C5,A6,00 10 3E,E5,10 0A
,A0 00 5C,0AF6
220 DATA12 40 04 5D 12 40 4C 73,
04 4C,A1 94 A0 10 A5,70 0C,00 03
,00 0C 03 071A
230 DATAA0 10 A5 00 0C,0F,03,0D
0E 03 A0 10 A5 C9,0C 11 03,00 10
 03 A5 35 0756
240 DATA00 11 0C 00 03,00 0A 03
A5,13 A0 10 00 00 03,0C,09 03 A0
 10 A5 55 062F
250 DATA00 02 03 0C,03 03,A9,6A,
A0 11 20 FC 20 A5,01,A0,21,04,2C
 05 20 A5 0700
260 DATA00 00 00 21,60 A5 00 03,
0D 20 73,04 C9,FE,F0,00 20 79,04
 4C 1E 94 0730
270 DATA20 73 04 C9 9C 00 50 C9,
A6 00 57 38,E9 00 0A A0 09,5C,12
 05 56 03 0A60
280 DATA50 12 06,55 20,54 00 4C,
17 93 93 45 56,54,45,4E 44,45 44
 20 42 41 083A
290 DATA53 49 43,20 46,4F,52 20,
54 48 45 20,43,4F,4D,4D,4F 44 4F
 52 45 20 05CC
300 DATA50 4C 55,53,2F,34,0D 42,
59 20 52 4F 42,45,50,54 20 4A,41
 55 43 4F 0503
310 DATA43 48 53 20 20 31 39 38
36 00 4C A1 94,4F 4C,C4,46,41 53
 D4 53 4C 0726
320 DATA4F D7,57 49,4E,44,4F,D7
4C 4F 4D 45 CD 40,49 4D 49,CD,4D
 45 52 47 0853
330 DATAC5 41,50,50 45,4E,C4 50,
4C,4F D4 57 52 49 54 C5 45 4E 94
 45 D2 44 0889
340 DATA4F 4D C5,51,55,49,D4 50
52 4F C3 44,50 52 4F,C3,45 50 52
,4F C3 43 050A
350 DATA47 4F 54 CF 43,47 4F,53
55 C2 50 4F D0 43,4D 41 49 CE 50
 4C 45 53 0889
360 DATAD4 56 41,52 4C 49,53,D4
44 55 4D D0 44 4D 45,52 47 C5,46
 49,4E C4 0984
370 DATA52 45 43,4F 52 44,A3 42
53 20 56 C5 44 45,53 C8 44 45 45
 C8 54 57 0942
380 DATACF 42 49 4E,A4 42 54,4F,
4D A4 48 54 4F 42 A4 45,56,41,CC
,41 53 CE 08F8
390 DATA41 43 D3 44,45 C7,52 41
C4 00 12 0A 12 A8,12,01 10,73 12
 CD 12 EA 00A7
400 DATA12 FC 13 E5,14 20,14,40
,14,51 14 01 14 57 14,50,0E 0A 0D
 02 15 0D 05C0
410 DATA15 16,15 3D 15 50 1A 53
16 60 1A 05,10 D3 1E 07 1E F7 1F
,53 1F 0E 05CE
420 DATA17 7F,10 00 15 03 13 3C,
15 40 10 00 15 05,1A,05 1A 30 1A
,1C 00 AD 04FE
430 DATA06 FF,20 EF 0D,06,FF,60,
4C 70 E3,A5 FF A0,01,51,28 20,10
 00 A5 22 0A42
440 DATA10 65,02,85 20 A5,23 65,
00 85 2E 4C 5A 0A 20,14,03,20 E4
 9D A4 15 07AA
450 DATAA5 14,84,2C,85,28,A9,00,
A0 00 51 28,E6 28 D0 02,E6 2C 4C
,78,0A 20 0804
460 DATA14 53,20,E4 9D,A4,15 A5,
14 84 30 85 37,60,4C,21,59 20 D6
,13 86 28 0851
470 DATA84 2C A0 00 30,51 20 20,
75 13 06 20,D6 2E 20 10 99 20 02
,13 A5 2E 075F
480 DATAA2,13 0D 02 03 8E 03 03
A9 01 85 3C A5 FF 05 30 A0,00 A9
,D0 20 54 0070
490 DATA04 05 D2,C0,A5,D0 20 94,
04 85 D3 F0 20 C0,A5 D0 20 94,04
 85 14 C0 0823
500 DATAA5 D0 20 94 04,85,15,A2
00 E8 C0 A9 D0 20 94 04 9D,FE,01
 D0 F4 0A 0830
510 DATAA0 20 34 07 A5 D2 A6 D3
85 D0 86 D1 D0 08 A9 12 A2,07 0D
 02 03 0E 00A0
520 DATA03 03 20 03 07 20 C3 13,
A5 00 85 AD A6 20,A4 2C 20 D5,FF
,85 10 20 07F6
530 DATA07,FF 29,0F F0 00 20 9D
13 A2 10 4C 03 06 60 48 20 9D,13
 60 4C 70,0523
540 DATAA7 A5 D0 38 E9,02 05,14,
A5 D1 E9 00 85,D1 A5 00 A0,01,14
,CD 91 14 0AF0
550 DATAA5 D2 A6,D3 85 20 86 2C,
A5 D0 A6 D1 85 20,06 2E 60 A5 00
 20 0D FF 0085
560 DATAA2 01 A0,00 20 0A FF 20,
0D A5 4C 7A A8,A5,20,85,D2 A5,2C
 85 D3 A6 0AE5
570 DATA2D A4,2E,90,D0,04 D1,60
20 D5 13 0A 38 E9 02 05 20,90,E9
 80 85 2C 09A1
580 DATA20 75,13 00 D0 04 D1,20,
02 13 20 10 00 60 20 14 93,20 E4
,9D A5 14 0079
590 DATA85 F1 A5 15 85 F2 20,91
94 20 14 93 20 E4 9D A5 15 A5,14
 A0 80 91 03F9
600 DATAF1,C8 0A,91,F1 90,4C 21,
99,20 0E 94 20 84 9D E0 20,00 03
,4C 1C 99 04A4
610 DATA0A 48 20 A5 A0 20 04 9D
E0 15 80 F1 60 A6,10 20 F0 FF,20
 00 94 80 0AF0
620 DATA20 29,14,4C 00 90 20,29,
14 4C 00 91,20,17,81 4C 00 C5 A2
 00 CE 30 0852
630 DATA00 02 C6 3C 20 73 04,F0,
10 C9 24 D0 06 20,A1 17,4C 70 14
,9D 00 02 0756
640 DATAE0,D0 E0,A5 00 9D 00 02,
A6 00,20 05 E3 00 A5 3C 91,7C 00
 A5 30,91 05A0
650 DATA7C 00 A5 3A 91 7C,00 A5
33 91,7C 00 A9,0D 91 7C,A5 20 85
 D0 A5 2C 0824
```

```
660 DATA05,D1,A0,00,A9,D0,20,94,
04,85,D2,C0,A9,D0,20,94,04,85,D3
,D0,05,A2,0046
670 DATA11,4C,83,06,A0,04,A9,D0,
20,94,04,C9,FE,D0,0A,C8,A9,D0,20
,94,04,C9,A43E
680 DATA0E,F0,0A,A5,D2,05,D0,A5,
D3,05,D1,D0,C9,A2,FF,E8,D0,A9,D0
,20,94,04,8E3D
690 DATAF0,11,C9,20,D0,00,20,C0,
17,4C,FE,14,D0,00,02,F0,E0,D0,DA
,D0,00,02,0A5D
700 DATA0B,D5,30,A5,D0,E9,01,85,
30,A5,D1,E9,00,85,3C,4C,DC,06,20
,14,93,20,0A86
710 DATAE4,9D,4C,50,0D,A0,05,20,
05,85,00,A5,3C,91,7C,00,A5,30,91
,7C,88,A5,0A15
720 DATA3A,31,7C,00,A5,39,91,7C,
88,A5,0D,91,7C,20,79,04,20,0E,15
,4C,DC,00,0515
730 DATAA5,FF,05,4A,A5,0D,05,02,
20,71,00,F0,05,A2,0C,4C,03,00,20
,69,A7,A9,0A15
740 DATA05,20,72,A7,60,A5,32,C5,
2E,D0,03,A5,31,C5,2D,D0,03,A5,00
,2C,A5,00,00DA
750 DATA05,3C,20,79,04,4C,00,15,
A5,0C,18,04,C6,30,C6,32,20,20,0D
,4C,DC,00,0742
760 DATA20,60,A0,EA,EA,EA,EA,20,
54,A5,A5,2D,85,5F,A9,2E,85,60,30
,A5,31,85,0AF9
770 DATA5A,E5,2F,05,D2,A5,32,85,
50,E5,30,85,D3,A5,33,38,E5,01,85
,51,A5,34,0A99
780 DATAE3,00,05,99,20,C7,00,A5,
37,85,41,A5,36,85,42,A5,50,85,D0
,05,37,E6,0A70
790 DATA85,A5,55,85,D1,05,30,A0,
20,A4,2C,A5,00,20,D5,FF,90,03,4C
,7D,A7,20,05C0
800 DATA07,FF,20,8F,F0,05,A2,1D,
4C,03,06,06,20,04,2E,06,5F,04,60
,A5,D0,05,0ACF
810 DATA5A,A5,D1,05,50,30,A5,33,
E9,01,A0,A5,34,E9,00,AA,90,30,E6
,5A,05,50,0AAA
820 DATAA0,0A,E5,50,AA,E0,90,F0,
1F,A5,5A,10,E5,50,05,5A,90,03,E6
,50,10,A5,0AEC
830 DATA5F,E5,50,05,5F,00,02,E6,
00,90,45,FF,A8,C8,C6,50,C6,60,A5
,5A,20,94,0026
840 DATA04,91,5F,C8,00,F6,E6,50,
E6,60,CA,D0,EF,38,A5,5F,05,31,E5
,D2,85,2F,0CEF
850 DATA45,60,45,32,E5,D3,05,30,
A5,41,85,37,A5,42,05,38,60,60,20
,10,00,A5,05A8
860 DATA00,20,90,FF,20,E7,FF,20,
D5,0A,20,EE,0A,4C,72,15,A5,9A,C9
,00,F0,01,0910
870 DATAC0,A5,20,05,AF,A5,2E,85,
60,A5,00,C5,30,D0,07,A5,AF,C5,2F
,D0,01,60,0600
880 DATAA4,00,A5,AF,20,94,04,C9,
00,00,6F,20,D2,FF,C8,A9,AF,20,94
,04,C9,7F,0829
890 DATA00,1A,20,D2,FF,20,6E,17,
A5,AF,A4,00,20,22,A2,20,6F,A4,20
,00,50,A5,0A00
900 DATAFF,F0,03,4C,4C,17,29,7F,
20,D2,FF,A9,24,20,D2,FF,20,6E,17
,A9,22,20,0988
910 DATAD2,FF,A0,00,A9,AF,20,94,
04,AA,F0,1E,C0,A5,AF,20,94,04,85
,22,C8,A9,0829
920 DATAAF,20,94,04,85,23,A9,00,
A9,22,20,94,04,20,D2,FF,C8,CA,D0
,F4,A5,22,0A14
930 DATA20,D2,FF,F0,51,D0,4F,00,
03,4C,70,16,29,7F,20,D2,FF,C0,A9
,AF,20,94,0B2E
940 DATA04,C9,7F,00,12,20,D2,FF,
20,6E,17,A9,46,20,D2,FF,A9,4E,20
,D2,FF,D0,0B3C
950 DATA29,20,7F,20,D2,FF,A9,25,
20,D2,FF,20,6E,17,A6,00,A9,AF,20
,94,04,85,0950
960 DATA62,C8,A9,AF,20,94,04,85,
63,A2,90,20,C9,A2,20,6F,A4,20,00
,A0,A9,00,0A40
970 DATA20,D2,FF,10,A5,AF,69,05,
85,AF,90,02,E6,90,20,E4,FF,20,E1
,FF,D0,01,00F0
980 DATA64,A5,C6,C9,10,D0,F1,30,
00,0F,A9,3D,20,D2,FF,10,A5,AF,69
,02,85,AF,00EE
990 DATA50,02,E6,00,00,20,73,04,
20,85,94,A5,13,40,A5,14,40,20,E4
,9D,A0,01,0890
1000 DATA01,14,AA,00,01,14,A0,60
,85,14,60,85,13,0A,4C,76,0A,A9,3
0,00,40,10,091A
1010 DATAA5,3C,0D,40,10,20,73,04
,C9,29,F0,03,4C,A8,17,20,73,04,D
0,01,60,A2,07E3
1020 DATA0E,4C,03,86,10,00,65,D0
,85,30,A5,D1,65,00,85,3C,20,D2,1
7,4C,33,10,0D10
1030 DATA20,73,04,20,40,10,00,42
,10,8C,43,10,A5,0E,0D,44,10,A5,0
D,8D,45,10,0620
1040 DATAA5,30,A4,3C,8D,46,10,8C
,47,10,AD,46,10,AC,48,10,85,30,8
4,3C,AD,42,001F
1050 DATA10,05,49,AD,43,10,05,4A
,20,73,04,20,2C,93,A5,30,8D,40,1
0,A5,3C,0D,076E
1060 DATA45,10,AD,44,10,05,0E,40
,AD,45,10,05,0D,40,A5,0E,40,A5,9
0,40,AD,46,001C
1070 DATA10,05,30,AD,47,10,05,3C
,60,20,73,04,C9,2C,F0,34,C9,29,F
0,03,4C,00,00A7
1080 DATA17,00,00,00,00,00,00,00
,00,00,00,A0,30,D0,02,C6,3C,C6,3
0,A2,00,24,04F3
1090 DATA40,0A,0A,E0,20,90,0E,20
,61,10,60,A9,00,05,00,20,73,04,4
C,A5,9C,4C,07D0
1100 DATA27,93,20,D2,17,4C,33,10
,20,04,9D,E0,10,00,3A,06,D0,20,D
0,9D,E0,20,0571
1110 DATA60,31,50,D1,20,D8,9D,E0
,20,00,20,00,D2,20,D0,9D,E0,10,0
0,1F,66,D3,0000
1120 DATA45,D0,C5,D3,00,17,A5,D1
,C5,D2,00,11,A5,D3,A9,D2,20,67,D
0,A5,D0,A6,0E12
1130 DATAD1,20,7A,DE,4C,91,D0,4C
,1C,99,A5,14,40,A5,15,40,20,73,0
4,20,85,94,0D02
1140 DATA20,67,9D,85,52,C9,10,F0
,07,C9,00,F0,03,4C,A1,94,A9,00,8
5,14,85,15,00EC
1150 DATA40,00,20,00,04,C0,C9,32
,00,1C,C8,30,50,18,0A,26,14,26,1
5,C4,52,D0,0865
1160 DATAE8,A5,14,A6,15,A0,68,85
,14,60,85,15,AA,4C,76,0A,4C,A1,9
4,A5,14,48,0872
1170 DATAA5,15,48,20,73,04,20,0E
,94,20,14,93,20,E4,9D,20,00,04,2
0,24,19,60,0747
1180 DATA85,15,60,85,14,4C,CA,9C
,A9,10,20,5C,50,A0,00,A9,00,85,1
4,26,15,2A,05E3
1190 DATA69,30,91,62,C0,C0,10,D0
,F0,60,20,3E,10,A5,14,48,A5,15,4
8,4C,10,05,05A0
1200 DATA20,73,04,20,05,94,20,10
,5E,A5,14,40,A5,15,40,20,E4,9D,2
0,24,19,60,0712
1210 DATA05,15,60,85,14,4C,CA,9C
,20,73,04,20,40,9C,C9,58,90,03,4
C,4C,CC,05,0005
1220 DATA0D,A5,30,48,A5,3C,48,A0
,00,20,00,04,99,00,02,C8,C4,D0,D
0,F5,A9,00,09FA
1230 DATA99,00,02,85,30,A9,02,85
,3C,20,53,05,20,14,93,60,05,3C,6
0,05,3E,60,0731
1240 DATA24,61,18,10,23,A5,61,29
,7F,C9,02,00,1C,C9,01,10,00,16,A
2,00,85,63,0787
1250 DATAC9,00,D0,0F,E8,E0,03,D0
,F5,A5,62,29,7F,C9,00,D0,02,30,6
0,4C,1C,99,0A10
1260 DATA20,73,04,20,05,94,4C,17
,93,20,CC,19,20,A0,19,90,03,4C,0
F,1A,A0,1A,0666
1270 DATAA2,30,20,55,A2,20,27,A6
,A0,1A,A9,30,20,9C,A0,A0,9F,A9,F
0,20,66,A0,0997
1280 DATA20,E4,A5,A0,1A,A9,30,20
,72,A0,4C,1A,A0,20,D5,19,A0,AA,A
9,EC,4C,EC,042C
1290 DATAA0,A5,66,10,06,20,7F,1A
,4C,27,A6,4C,7F,1A,20,CC,19,A9,3
9,A0,91,20,0703
1300 DATA3C,A0,20,94,A2,A0,1A,A0
,33,20,21,A2,4C,97,A1,66,34,00,0
0,00,00,00,0780
1310 DATA00,00,00,20,CC,19,A0,1A
,A5,33,20,5C,A0,20,04,A2,A5,39,A
0,94,20,21,076A
1320 DATAA2,4C,97,A1,20,E7,FF,A9
,00,20,00,FF,A5,04,AA,A0,FF,20,0
A,FF,20,05,0C25
1330 DATA27,A2,04,20,97,A7,A9,0D
,20,D2,FF,20,79,04,20,FF,0A,A2,0
4,20,C3,FF,0A20
1340 DATA4C,E7,FF,A9,EC,A0,AA,4C
,21,A2,20,94,9D,E0,02,00,03,4C,1
C,99,66,E0,003D
1350 DATA49,04,AA,A0,FF,20,0A,FF
,A9,00,20,00,FF,20,C0,FF,A2,04,2
0,C9,FF,A6,0C67
1360 DATAE0,F0,03,4C,00,1C,AD,13
,FF,C9,D1,F0,00,A9,11,0D,FE,07,2
0,D2,FF,4C,00D3
1370 DATAC0,1A,A9,91,0D,FE,07,20
,D2,FF,A5,04,A5,22,A9,0C,85,23,A
2,00,A0,50,059E
1380 DATA90,05,40,30,01,22,65,5E
,85,5D,38,50,46,A5,40,A0,A5,5E,2
9,00,F0,09,090D
1390 DATA20,09,10,A5,5E,23,7F,85
,5E,A5,5E,E9,20,10,10,00,69,60,9
9,AC,10,4C,0803
1400 DATA2F,10,69,20,30,E9,40,10
,10,00,69,40,99,AC,10,4C,2F,10,6
2,40,30,E9,06CD
1410 DATA60,10,10,00,69,60,99,AC
,10,4C,2F,10,69,00,30,E5,00,10,1
0,05,69,C0,07EF
1420 DATA99,AC,10,60,A0,C0,E6,40
,A5,5D,29,80,F0,03,20,90,10,C0,1
4,D0,05,A0,0A87
```

```
1430 DATA00,89,AC,10,20,D2,FF,CA
,C4,48,D0,F5,E0,E0,32,D0,12,A9,9
1,20,D2,FF,0D14
1440 DATAA9,0D,20,D2,FF,A9,84,20
,C3,FF,4C,CC,FF,10,A5,22,69,14,0
5,22,A5,23,0A10
1450 DATA69,00,85,23,A5,22,20,04
,FD,03,4C,D2,1A,A9,0D,20,D2,FF,A
0,FF,07,20,00A8
1460 DATAD2,FF,4C,D2,1A,0A,85,4C
,A5,48,AA,A9,12,9D,AC,10,E6,48,A
5,4C,AA,C0,0801
1470 DATA60,0A,00,4C,A5,48,AA,A9
,52,9D,AC,10,E6,48,A5,4C,AA,60,0
0,00,00,00,0520
1480 DATA00,00,00,00,00,00,00,00
,00,00,00,00,00,00,00,00,00,00,00
,00,00,00,0000
1490 DATA00,00,00,00,00,00,00,00
,00,00,00,00,00,00,20,92,1C,A9,0
6,A0,1C,20,02D9
1500 DATA00,00,20,72,F1,A0,02,03
,B0,91,1C,A0,03,03,B0,91,1C,A9,0
E,20,A6,1C,090C
1510 DATA05,AC,00,85,1C,A9,00,00
,AD,20,C0,FF,AE,06,1C,20,C6,FF,A
9,04,0D,2A0831
1520 DATA03,A9,1C,00,20,03,A9,1E
,00,02,03,A9,1C,00,03,03,20,CF,F
F,20,CF,FF,0910
1530 DATA20,CF,FF,05,14,20,CF,FF
,05,15,05,14,F0,36,A5,00,D0,32,2
0,CF,FF,05,0AF0
1540 DATA14,20,CF,FF,05,15,A0,00
,20,CF,FF,99,00,02,AE,F6,07,E0,3
F,F0,15,C8,0A60
1550 DATAC5,00,D0,EE,50,04,00,A9
,50,D0,00,A9,02,A0,00,85,3C,84,3
0,A4,00,4C,0507
1560 DATA36,07,AD,00,1C,0D,02,03
,AD,01,1C,0D,03,03,A5,00,0D,2A,0
3,A9,EF,0D,0025
1570 DATA20,03,AD,00,1C,20,C3,FF
,20,CC,FF,A5,00,07,60,00,4D,45,5
2,47,45,4E,0941
1580 DATA47,20,00,00,00,00,A5,E6
,20,21,C0,20,05,CC,A5,00,0D,70,0
2,05,0A,A0,0782
1590 DATA85,3C,3F,CA,A0,97,E0,00
,F0,0E,D0,00,05,D0,00,30,E9,01,4
C,A6,1C,CA,092F
1600 DATA00,F2,60,20,0F,C7,A5,00
,20,D2,FF,A5,00,20,D2,FF,A5,00,0
D,AF,10,A5,000A
1610 DATA06,0D,00,10,A5,00,00,AC
,10,0D,AD,10,A5,00,00,01,10,AD,A
F,10,0D,AE,05E4
1620 DATA10,AD,AC,10,0D,02,10,AD
,AD,10,29,F0,0D,03,10,A9,00,0D,0
4,10,0D,05,0A21
1630 DATA10,AD,AE,10,4A,4A,4A,4A
,F0,17,A5,00,A0,40,10,50,05,40,A
0,50,69,01,000C
1640 DATA90,CA,D0,F4,0C,05,10,60
,60,04,10,AD,AE,10,20,07,00,06,1
0,AD,AD,10,0A6F
1650 DATA29,07,0D,07,10,A5,07,30
,ED,07,10,0D,07,10,10,A5,00,0D,0
5,10,A0,A5,08E4
1660 DATA20,6D,04,10,AA,D0,0D,03
,10,A0,0A,50,02,10,AA,50,50,00,1
0,0D,05,10,0A26
1670 DATA0A,69,00,0D,00,10,AC,05
,10,AD,00,10,05,D1,A5,00,00,D0,A
9,D0,20,94,0AD4
1680 DATA04,0D,0A,10,A9,01,AE,07
,10,F0,04,0A,CA,D0,FC,20,0A,10,F
0,15,AD,AE,0A06
1690 DATA10,30,ED,AF,10,80,00,10
,A9,01,AE,00,10,F0,04,0A,CA,D0,F
C,10,50,01,0A65
1700 DATA10,0D,01,10,AD,AF,10,10
,60,00,10,EE,AE,10,CD,AE,10,30,0
3,4C,E0,1C,030E
1710 DATAAD,01,10,20,D2,FF,10,AD
,AD,10,69,01,8D,AD,10,AD,AC,10,0
9,00,0D,AC,09CC
1720 DATA10,A5,01,CD,AC,10,D0,07
,A5,48,CD,AD,10,F0,03,4C,DA,1C,A
9,0D,20,D2,0988
1730 DATAFF,AD,AF,10,10,69,07,0D
,AF,10,C9,C4,90,03,4C,D2,1C,C9,C
0,F0,08,A5,00A4
1740 DATA03,8D,00,10,4C,D2,1C,A5
,0F,20,D2,FF,A5,0D,20,D2,FF,20,D
2,FF,A5,04,0A63
1750 DATA20,C3,FF,4C,CC,FF,20,DE
,05,A5,30,0D,33,1E,00,4E,1E,A5,2
0,05,5F,A9,0A0D
1760 DATA2C,00,00,A0,01,04,0F,20
,6D,01,F0,43,20,E4,FF,C9,03,F0,3
C,A0,03,A2,05CC
1770 DATAFF,04,03,E0,00,02,02,F0
,33,C9,20,F0,F6,C9,22,F0,F2,C0,2
,60,01,F0,0CB4
1780 DATA12,C9,20,F0,F0,C9,02,F0
,F2,50,02,2F,F0,DF,A4,03,C0,D0,D
0,A4,00,20,0003
1790 DATA00,01,AA,C0,20,60,01,00
,5F,00,60,D0,04,4C,C0,1E,A0,01,0
4,0F,20,3E,0503
1800 DATA00,C0,20,60,01,AA,C0,20
,60,01,04,49,20,5F,A4,A9,20,A4,4
9,20,7F,20,0554
1810 DATA02,00,C9,02,D0,00,A9,0F
,49,FF,85,0F,C0,F0,02,20,60,01,F
0,00,10,E9,0801
1820 DATAC9,FF,F0,E5,C9,FE,F0,3A
,D0,47,24,0F,30,D0,AA,04,49,A0,0
0,0A,F0,10,0C04
1830 DATACA,10,0C,E6,22,D0,02,E6
,23,01,22,10,F6,30,F1,C0,01,22,3
0,00,20,0210
1840 DATA50,D0,F0,20,3E,00,A0,01
,A9,00,91,30,00,A9,3F,91,30,A2,0
0,4C,03,00,0400
1850 DATAA2,11,86,23,A2,AF,86,22
,C0,20,60,01,4C,A6,1E,A2,01,86,2
3,A2,0E,06,0580
1860 DATA22,4C,A6,1E,20,04,90,E0
,3F,90,03,4C,1C,55,86,D0,20,51,9
4,20,14,53,0BE0
1870 DATA20,E4,90,04,D1,05,D2,20
,12,1F,4F,06,13,20,97,A7,10,A5,0
0,69,60,03CC
1880 DATA00,50,1F,A5,01,80,51,1F
,A5,02,00,52,1F,A5,03,00,53,1F,A
0,00,85,4F,0A63
1890 DATA1F,20,02,50,C0,C8,00,D0
,F5,4C,FE,90,A9,00,80,03,20,30,A
0,20,00,90,0A00
1900 DATA86,D3,00,50,00,00,00,00
,20,60,A6,A5,20,A4,2C,85,D0,84,D
1,A5,20,A4,00FC
1910 DATA2E,05,D2,04,03,20,7E,1F
,20,D0,FF,A5,D0,A4,D1,85,20,04,2
C,A5,D2,A4,00F5
1920 DATA03,85,20,04,2E,60,20,30
,1F,85,2C,A4,20,20,00,1F,40,98,A
A,60,A6,A5,0ADF
1930 DATA20,60,20,9D,A6,20,91,54
,20,14,53,4C,E4,9D,EA,EA,EA,20,4
8,5C,85,D0,0AE0
1940 DATA00,D1,04,D2,20,F7,1F,A0
,00,C4,57,F0,00,09,10,05,C9,6F,F
0,1F,C9,4C,080F
1950 DATAAD,1F,A9,0F,20,A6,1C,85
,D4,A0,0F,A6,D3,20,8A,FF,A9,00,2
0,0D,FF,20,0A65
1960 DATA85,A7,A6,D4,4C,E1,1F,B9
,13,05,C5,D3,D0,DA,0E,09,05,20,9
7,A7,A5,00,00A4
1970 DATA06,D1,86,22,A4,D2,84,23
,20,0E,90,20,3E,90,4C,FE,00,20,7
9,04,00,05,0504
1980 DATAA2,00,86,D3,60,20,D9,90
,E0,00,00,F6,4C,1C,55,CC,06,25,0
9,00,02,04,0703
1990 DATA12,00,09,00,12,15,2C,05
,00,12,1E,2F,47,7A,7E,0F,DC,E0,E
9,F3,FD,09,0750
2000 DATA13,10,34,5C,61,5C,5E,60
,62,64,66,68,6A,6C,6E,70,72,74,7
6,7C,7E,00,0857
2010 DATA02,04,06,00,0A,0C,0E,50
,52,54,56,90,9A,9C,9E,A0,A2,A4,A
6,FF,00,15,0C10
2020 DATA19,77,0E,90,E0,F6,F0,4E
,54,71,74,EE,F1,3A,71,60,A5,05,C
6,01,16,2F,0077
2030 DATA05,AA,04,CE,01,D7,DA,DD
,E2,E7,EE,F1,F4,F7,FE,03,10,15,1
0,1E,2A,2F,0D70
2040 DATA00,5F,70,73,1A,3E,5C,D7
,DA,DF,E1,E0,FC,07,15,10,1E,2A,3
F,41,A0,0F,03F9
2050 DATAEC,FE,01,0E,11,1E,21,2E
,3C,45,7A,00,53,A5,06,DA,E6,EC,F
1,F6,00,0A,0AA5
2060 DATA14,64,6A,7A,86,CC,01,06
,09,0E,E1,E4,E7,EA,ED,F2,F7,FA,F
D,16,1A,1D,0EEC
2070 DATA22,25,2A,30,33,35,3F,44
,48,4E,51,57,5A,5D,60,70,79,7E,9
2,05,0A,94,077D
2080 DATA97,9A,9E,A1,A4,A8,AC,03
,00,00,C8,C5,CC,01,D9,DF,E6,EE,F
2,0F,12,69,0E0A
2090 DATA0F,EC,F7,13,22,27,2C,31
,36,69,AA,00,C0,D6,20,60,90,A0,0
0,05,00,21,05D4
2100 DATA99,00,06,C0,C0,00,D0,F5
,60,70,80,3F,FF,05,D2,C9,10,D0,1
5,A9,10,00,007F
2110 DATA46,06,A9,69,00,47,06,A9
,20,00,35,00,00,3C,06,4C,31,06,A
9,30,00,45,05CF
2120 DATA06,A9,E9,00,47,06,A9,50
,00,35,00,00,3C,06,A2,00,00,07,2
0,00,D0,A0,0001
2130 DATA00,09,10,20,05,D1,C0,04
,03,A0,00,01,D1,10,65,30,91,D1,A
4,03,C4,D0,00A9
2140 DATA00,E9,10,90,60,30,06,80
,30,06,90,03,EE,3C,06,E0,E6,D2,E
0,10,D0,CE,00D6
2150 DATAA5,D2,C9,40,00,03,A2,40
,2C,A2,10,0E,00,03,06,0F,03,0E,1
1,03,0E,05,076A
2160 DATAD3,0E,03,03,0E,60,13,E0
,0E,06,03,A9,10,00,3F,06,00,3E,F
0,50,60,00,0730
```

```
PROGRAM  +4 EX BASIC 2

10 REM L1S1NG2
20 SCNCLR
```

```
30 WRITE#2,1)"OD YOU OWN A PRINT
ER ",,OO GETP$ LOOPUNTILP$="Y"OR
P$= N" PRINTP$
40 WRITE#2 3)"OO YOU OWN A DISKO
RIVE ",,OO GETO$ LOOPUNTILO$="Y"
ORD$="N" PRINTO$
50 PRINT"ICLEAR)CHA1N"+CHR$(34)+
"L1STING3 "+CHR$(34)+" ", 1FD$="Y
"THENPRINT"8" ELSEPRINT"1"
70 POKE1319,15 POKE1320 13 POKE2
35 2 END
```

PROGRAM +4 EX BASIC 3

```
10 ORAPH1C1,1
20 REM L1ST1NG3
30 DEFFNK(X)=OEC1"2000")+1NT1Y/0
)+320+Y-1NT1Y/0)*8+1NT1X/0)*8
40 CHAR1,5,0 "GRAPH1C DEMO"
50 C1RCLE1,168,100,10
60 FOR1=20TO70STEP2 C1RCLE1,100,
100,1 NEXT
70 PA1NT1,171,100
80 X=0 Y=0
90 FOR1=1TO49
100 READA$
110 A$=R1GHT$(HTO0$1A0),8)
120 FORT=1TOLEN1A$)
130 Q=VAL(M1O$(A$,T,1))
140 POKEFNK(X) PEEX(PNK(X))OR2 1
8-T)*Q
150 X=X+1
160 NEXT
170 1FX=24THENX=0 Y=Y+1
180 NEXT
185 1FP$= Y"THENDUMP1
200 GEIKEYY0 ORAPH1C0
210 DATAFF,FF,FF,FF,FF,C0,00,
03,C0,30,03,C0,0C,03,C0,44,03,C0
,6C,03,C5,3A,03
220 DATAC5,00,03,OF,,FF,FF,C0,00,
03,OF,FF,FF,C0,00,03,OF,FF,FF,C0
,00,03,OF,FF,FF
230 DATAC0,00,03,OF,,FF,FF,C0,00
,03,OF,FF,FF,C0,00,03,C0,00,03,FF
,FF,FF,FF,FF,FF
240 PRINT"[CLEAR]" 1FD$="Y"THEN
PRINT DMERGE"+CHR$(34)+"L19T1MO4
+CHR$(34)
250 1FD$="N THENPRINT"MERGE"+CHR
$(34)+ L19T1NO4"+CHR$(34)
260 POKE1534,ASC1F$) POKE1535,A9
C10$)
265 POKE1319,19 POKE1326,13 POKE
1321,02 POKE1322,213 POKE1323,13
270 POKE235,5 END
```

PROGRAM +4 EX BASIC 4

```
15 REM L1ST1NG4
25 GOTO30*10
300 GRAPH1C0,1
310 WRITE#14,0)"LO-RES DEMO"
320 WRITE#14,1)"[c U]Ic U]Ic U]I
c U]Ic U]Ic U]Ic U]Ic U]Ic U]Ic
U]Ic U]"
330 PRINT"[c A]Ic R]Ic R]Ic R]Ic
 R]Ic R]Ic R]Ic R]Ic R]Ic R]Ic R
]Ic R]Ic R]Ic R]Ic R]Ic R]Ic R]I
c R]Ic R]Ic R]Ic R]Ic R]Ic R]Ic
R]Ic R]Ic R]Ic R]Ic R]Ic R]Ic R]
Ic R]Ic R]Ic R]Ic R]Ic R]Ic R]Ic
 R]Ic R]Ic R]Ic 9J"
340 PRINT"1m 0]Im 0]Im 0]Im 0]Im
 0]Im 0]Im 0]Im 0]Im 0]Im 0]Im 0
]Im 0]Im 0]Im 0]Im 0]Im 0]Im 0]Im
 0]Im 0]Im 0]Im 0]Im 0]Im 0]Im 0]
```

Word Processor Roundup

As a word processor is defined as a simple means of entering, editing, printing, and storing text on a computer why are there so many of them? This article covers no less than twelve different word processors, each offering a variety of functions and options to cater for the incredible variety of uses these essential programmes are used for.

By Tony Hetherington

If you have ever written a letter memorandum or magazine article then you need a word processor and so you will no longer be buried in early drafts and pools of correcting fluid. With a word processor you can type on the keyboard anything you would on a typewriter and then correct it, alter it, save it for later use and print it out.

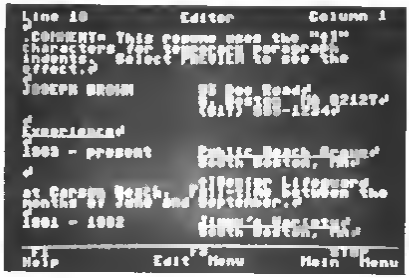
In this article (which I wrote using PaperClip) I have looked at twelve word processors and assembled their strengths and weaknesses and included certain factors and features that may help you find the word processor that is best suited for your needs. (Table 2)

EASY SCRIPT/Commodore

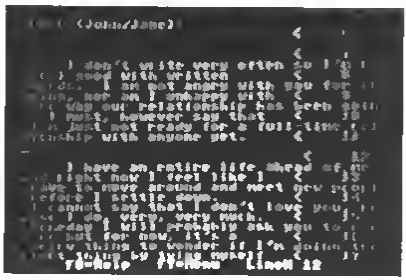
Easy Script is probably one of the best known C64 word processors as it was given away free with the 1541 disk drive.

Although Easy Script contains facilities to create and edit documents, it uses a system of control commands that means that text on the screen isn't the same as it will appear on the paper. In fact the program includes a special function to show the document as it will actually appear. Since then the word processors have become friendlier which means you can get started without wading through a huge manual.

• *Ski Writer*



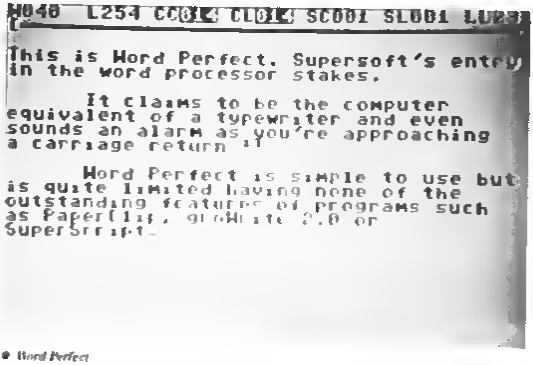
• *Master Word*



SKI WRITER/Mastertronic

Mastertronics imported Ski Writer includes only the briefest of manuals (three pages of a minute leaflet) and consists of four main functions that are assessed through a main menu

The Edit option allows you to type and edit your document and format it using Easy Script style commands to set margins page numbers headers and text justification The Preview mode interprets these commands and displays the text as it would appear on paper Your document can then be saved loaded or merged from disks with the file option that can also format blank disks before printing on any defined printer and paper type



● *Word Perfect*

MASTER WORD/Argus Press Software

As part of the Load N Go series of cheap and cheerful disk programs Master Word has the distinction of being the only word processor that is supplied with no instructions at all!

If this complete lack of documentation hasn't put you off you'll find pages and pages of help screens to get you going as well as files of sample letters which include nine different business letters nine letters home five true love notes two the romance is over and one it never really began letter for the one you hate!

If you don't expect too much from Master Word you won't be disappointed

WORD PERFECT/Supersoft

Word Perfect is an easy to use program but has its limitations including a 254 line limit to documents This is just over four A4 pages and so limits its applications to short notes letters and memos Longer articles such as this one just wouldn't be possible

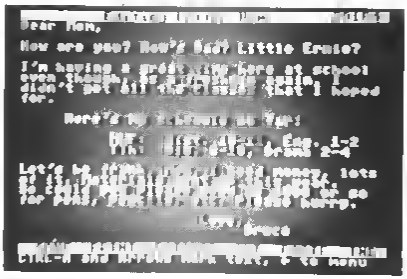
If you stay within this limitation you'll be able to enter and edit text using simple commands that try to mimic a typewriter You even hear a bell when your e hearing a carriage return although you can let the program automatically wordwrap to a new line

CUT & PASTE/Ariolasoft (Electronic Arts)

Many word processors claim to be easy to use but this one actually is so much so that you can load in a sample document, edit it, add in your own words of wisdom and save it without looking at the slim introduction manual!

Some word processors include a command bar, Cut & Paste has two! At

● *Cut & Paste*



● *Homeword*

the top of the screen you get a menu of files on disk and at the bottom a duck shoot command bar to load and save these files and format blank disks

Once your re actually editing text whether it s one of the supplied format ted letters (to Mom asking for money!) resume or memos or all your own work then cut and paste is the name of the game Any word sentence or block of text can be defined and cut from the document and then pasted in another part of it

This means you can move and copy text as well as restore deleted words that you really wanted (as long as you haven t cut anything else or even cut a general introduction that can then be pasted into a whole series of different documents

HOMEWORD/Green valley

Homeword (also sold as Master Writer) is the only icon controlled word processor in this collection It has a picture of a filing cabinet for its load merge and save options as well as a printer a page of text for move copy erase and find and replace edit options a layout icon to set text justification and a disk to catalog and create files

Selecting any icon leads to another set of icons and so on until you find the option you need As with Cut and Paste the manual is almost redundant as it s always clear through icons and screen prompts what you re doing, how you got there and how to get out of it without losing all your work

To help you further a display at the bottom of the text entry screen shows the current page a bar illustrates the remaining memory and a bit pattern representation of your page is shown

TEAM-MATE/Tri Micro

Team Mate is in fact the C64 version of the programmes supplied with Plus/4 and features an integrated package of a word processor spreadsheet database and graphics package (that produces bar and pie charts)

The word processing part of the package is undoubtedly limited with each file restricted to only 99 lines However more than one file can be linked together to form longer documents that can use the output from graphics and spreadsheet packages in its documents It doesn t compare with the full word processors but is a useful addition to a package

TRI WORD/

Tri Word is part of the Triangle (word processor/spreadsheet and file handler package) and is less limited than its Team Mate competitor with up to 400 lines in a file However data cannot be mixed between the programmes and files can t be linked together However you shouldn t need to as the files should be big enough One thing that remains a mystery is why F1 moves the page down and F7 moves it up!

MINI OFFICE II/Database

The Mini Office II word processor is probably the best word processor supplied in an integrated package

From a single menu you can edit and create your text preview it search and replace consistent spelling errors and load and save your work Meanwhile a bar at the top of the screen keeps track of the various modes you are in the time it has taken you to create the document and an extremely useful word count In fact everything you'd expect from an ordinary word processor which you get with a spreadsheet, database graphics package label printer and communications package!

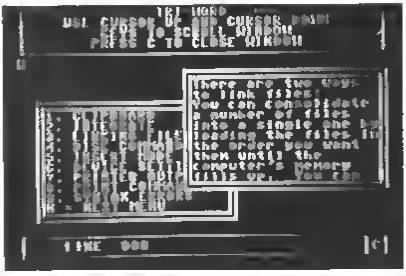
GEOWRITE 2.0/First Analytical

GeoWrite 2.0 is just part of the Writer s Workshop package that runs with the pull down menus and pointers of the Mac like GEOS operating system

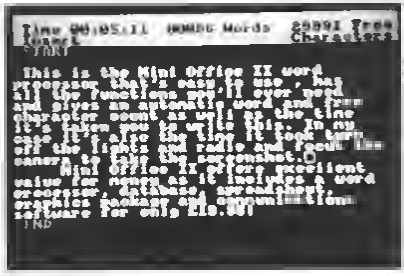
A version of geoWrite was included with the GEOS master disk but this was little more than a text handler and has been replaced by a full word processor featuring headers and footers left right and centre justification plain bold italic outline subscript and superscript text as well as a choice of fonts in a variety of point sizes and the incredibly powerful text grabber

The text grabber can convert any C64 word processor document into

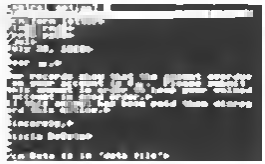
● *Tri Word*



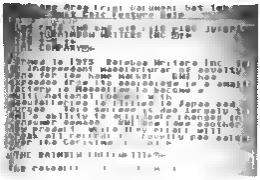
● *Mini Office II*



● Creatrix



● Paperclip



● Superscript

genWrite format which can then be edited by genWrite and improved by adding different fonts and genPaint graphics

PAPERCLIP/Ariolasoft (Batteries Included)

I used Paperclip to write this article because I found it the easiest machine to use. I needed a word processor that didn't have any complicated control codes that you could accept type directly onto the screen and could easily correct mistakes as you go. Also which had a spell checker to remove any spelling mistakes before being saved to disk for direct entry into a typesetting machine.

I didn't want to be restricted to a small number of lines but I needed to know how many words I had used. (The Spellchecker is only available in the special Paperclip with Spell Checker pack which is definitely worth the extra fiver.)

The spell checker is the secret of Paperclip's success as it compares these words with its 15,000 word dictionary and prompts you to alter any it can't find or understand. Therefore it can spot typing errors, spelling mistakes and words run together through missed spaces, while giving you a report of the total number of words used and their average length.

This time I didn't need any of its other features such as transferring text between different documents, sorting capabilities, editing data from databases and spreadsheets or the ability to create form letters. But I did use the special commands to construct the comparism table (Table 2).

Paperclip with the added spell checker is more expensive than the others but you do get more for your money. Great value and a must for journalists with deadlines!

SUPERSCRIPT/Precision

I may prefer Paperclip but there are others such as Your Commodore's illustrious Editor who prefers Superscript to create his words of wisdom.

Superscript is undoubtedly an incredibly powerful word processor that's equal to Paperclip in the feature it offers including a spell-checker and mail-merge facility.

If there's any difference then it's how the commands are accessed. In Paperclip these are through conversational two key commands whereas Superscript employs a series of nested duckshoot menus. For example, selecting Document from the main menu leads to a sub menu that includes options to load, save, directory and spell. Select spell and you're led to another menu and options to check spelling, view or print the user dictionary and display a word count.

Comparing Superscript and Paperclip is like the comparism between a Ferrari and a Porsche. They are both equally impressive but different to drive.

The following table compares the tested word processors with eight factors that are important to a buyer. The final two factors are an opinion of the programmes themselves and their manuals and take into account friendliness and general ease of use.

	Easy Script	Ski Writer	Master Word	Word Perfect	Cut & Paste	Home Word
Headers	Yes	Yes	No	Nil	Yes	Yes
Footers	Yes	Nil	Yes	No	No	Yes
Line Spacing	Yes	Yes	Yes	No	Yes	Yes
Search/Replace	Yes	Nil	No	No	No	Yes
Help Function	No	No	Yes	No	No	Nil
Spell Checker	No	Nil	No	No	No	No
Word Count	No	No	Nil	No	No	No
Price		£4.99		£19.95		
Program	Fair	Fair	Good	Fair	Good	Good
Manual	Poor	Poor	None	Fair	Good	Good

	Team Mate	Tri Word	Mini Office	Geo Write 2	Paper Clip	Super Script
Headers	No	Yes	Yes	Yes	Yes	Yes
Footers	No	No	Yes	Yes	Yes	Yes
Line Spacing	Yes	No	Yes	Yes	Yes	Yes
Search/Replace	Yes	Yes	No	Yes	Yes	Yes
Help Function	No	Yes	No	Yes	No	Yes
Spell Checker	No	No	No	No	Nil	Yes
Word Count	Nil	No	Yes	No	Yes	Yes
Price			£19.95	£37.50	£44.95	£39.95
Program	Fair	Fair	Good	Good	Great	Great
Manual	Fair	Fair	Good	Good	Good	Good

Everyman's Guide to Graphics

Graphics are a fascinating application for the C64. In this comprehensive guide, we point the way to better understanding and use of this facility.

By Allen Webb

In my view the crucial part of any piece of software is the graphics. There are very few items which need no attention to graphics with even a text only package being improved by a redesigned font.

In this article, I want to give a detailed run down of the C64's graphics capability and how you can use it. Where it simplifies life I will give listings of helpful routines.

Vic Chip

First let us consider the driving force behind graphics, the VIC-II chip. This chip controls the graphics system which can in turn be altered via a number of registers. These registers are memory mapped allowing you to change them easily. Table 1 lists the most useful registers.

That's a pretty meaty lump of information and it's only provided as reference material. The rest of this piece will show you how the more important registers are used.

If you want to use your 64 efficiently, an appreciation of how it handles its memory is necessary. Figure 1 gives a simple memory map. The memory map can be considered to consist of two layers. The bottom layer is a block of 64K of RAM. On top of this are superimposed two areas of ROM and other chips. Since different devices occupy the same addresses a register at address one is used to decide

Table 1

Register	Bit	Function
		Sprite positions
$D000-$D010 (53248-53264)		
$D011 (53265)	Bit	
	7	Raster Compare
	6	Extended colour mode
	5	Bit map mode
	4	Blank screen
	3	24/25 row text
	2-0	smooth scroll Y direction
$D012 (53266)		Raster Read/write
$D013-$D014 (53267-53268)		Light Pen registers
$D016 (53270)	4	Multicolour mode
	3	38/40 column text
	2-0	Smooth scroll X direction
$D017 (53271)		Y expand register
$D018 (53272)		Memory Control Register
	7-4	screen matrix
	3-1	Character table
$D019 (53273)		Interrupt register
$D01A (53274)		IRQ mask register
$D01B (53275)		Sprite priority register
$D01C (53276)		Sprite colour mode register
$D 01D (53277)		X expand register
$D01E (53278)		Sprite to sprite collision register
$D01F (53279)		Sprite to background collision register
$D020 (53280)		Screen border colour
$D021 $D024 (53281-53284)		Background colour registers
$D025-$D026 (53285-53286)		Sprite multicolour registers
$D027-$D02E (53287-53294)		Sprite colour registers

which are switched in In normal use, the RAM under the ROMs is unavailable to Basic but it can be used for graphics

The 64 treats the block of RAM as four banks of 16k

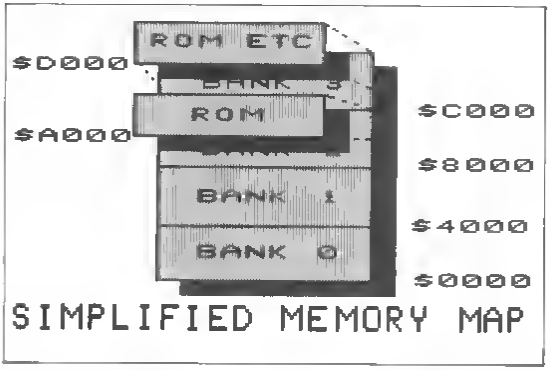
Bank0–$0000–$3FFF(0 to 16383)
Bank1–$4000–$7FFF(16384 to 32767)
Bank2–$8000–$BFFF(32768 to49151)
Bank3–$C000–$FFFF(49152 to 65535

Bank 0 is the default bank The bank in use is specified in bits 0 and 1 of location $DD00

The VIC can only address one bank at a time and it expects to find an area of screen memory and a character set within the bank This approach offers almost unlimited flexibility but also makes the use of graphics in the default bank restricted

Since the CPU and the VIC chip operate independently, the CPU doesn't care which bank is used for graphics We can therefore reconfigure the machine from Basic very easily

Let us consider how to reconfigure the memory map



Changing the BANK

This is achieved easily by changing the register at $DD00

10 POKE 56578 PEEK(56578) OR 3
20 POKE 56576, (PEEK(56576) AND 252) OR(3-BN)

Line 10 prepares the ground and line 20 switches in BANK number BN

The VIC chip ignores the absolute address of the bank and uses only the relative addresses within the bank I e each bank ranges from $0000 to $3FFF

Moving the Character Set

The register at 53272 tells the VIC chip where to get its character data In fact bits one to three hold this information

This information is changed by

POKE 53272 (PEEK(53272) AND240) OR X

X is equal to the start adress of the character data divided by 1024 With only three bits used only eight character sets are possible I e X = 0,2,4,6,8,10,12,14

Since the machine powers up with a character set there must be default information somewhere In fact, the default character set is held in ROM This data is imaged to banks 0 and 2 and is found at the following addresses

$1000-$17FF (Lower case set X=4)
$1800-$1FFF (Upper case set X=6)

Clearly it is possible to have a number of different sets of characters in a bank and simply switch between them as needed

Moving the Screen

The screen comprises of 1000 bytes of contiguous memory which usually resides between locations 1024 and 2024 This position is specified in bytes 4 to 7 in location 53272 These bytes actually specify the position of the screen in any bank of memory and can be changed by

POKE 53272 (PEEK(53272) AND15) OR Y

Y is equal to the start address of the screen divided by 64 This time we have four bits in the register allowing 16 possible screen positions with Y ranging from 0 to 240 in increments of 16 Unfortunately you cannot use all RAM areas for the screen If you use the areas imaged by the character ROM you will get garbage on the screen

In addition to changing the VIC register you must also tell the operating system where the screen is This is done with

POKE 648 SCREEN/256

where SCREEN is the start address of the screen area

The screen colour matrix cannot be moved and in fact presents no difficulties

Listing 1 allows you to reconfigure your 64 The first part asks you to specify where the screen and character set are to go These values are checked to ensure that they are in the same bank and are not at the same address It doesn t check any further so beware Line 60 to 80 calculate the register values Line 90 checks to see if you need to copy down the character set and lines 100 to 150 do this job if required Lines 160 to 190 reconfigure the machine

Listing 1 Reconfigure

```
26  10 PRINCHR$(147)  INPUT  SC
    REEN POSITION  SCREEN
44  20 INPUT  CHARACTER SET ADDR
    ESS' CHARS
```

Table 2		
Pixel one	**Pixel two**	**Colour Register**
clear	clear	53281
clear	set	53282
set	clear	53283
set	set	colour matrix

```
FD   30 IF SCREEN = CHARS THEN E$
     = ERROR  CHARS AND SCREEN AT
      SAME ADDRESS" GOTO60000
A1   40 IF INT(SCREEN/16384)=INT(
     CHARS/16384)THEN60
34   50 E$= ERROR  CHARS AND SCRE
     EN NOT IN SAME BANK  GOTO600
     00
1B   60 X=INT(SCREEN/16384)
5B   70 Y=(SCREEN-X*16384)/64
95   80 Z=(CHARS-X*16384)/1024
E1   90 IF Z<>4 AND Z<>6 THEN
BE   100 PRINTCHR$(147) COPYING R
     OM CHARACTER SET   THIS WILL
     TAKE A WHILE"
F4   110 POKE56334 PEEK(56334)AND
     254
4F   120 POKE 1 PEEK(1) AND 251
A2   130 FOR I=0TO2047 POKECHARS+I
      PEEK(53248+I) NEXT
F6   140 POKE1 PEEK(1)OR4
03   150 POKE56334 PEEK(56334)OR1

17   160 POKE56578 PEEK(56578)OR3

12   170 POKE56576 (PEEK(56576)AN
     D252)OR(3 X)
38   180 POKE53272 Y*X
B3   190 POKE648 SCREEN/256
C3   200 PRINTCHR$(147) DONE"
52   210 END
E0   60000 PRINT PRINTER
9C   60010 GET (# 1,A$:A$=""THEN600I
     0
38   60020 GOTO10
```

In my view the crucial part of any piece

Run Listing 1 putting the screen at 50176 and the character table at 51200 and then enter the line

POKE 44 4 POKE 1024 0 NEW

You will have a machine offering 39933 bytes for Basic and 144 sprites That s a lot more than you get on switch on! This extra capacity is achieved by

1) Using BANK 3 and moving the screen and character set to a handy block of RAM between the ROMs

2) Moving the start of Basic program storge down to 1025 Since we've moved the screen we can use the normal screen area for Basic

3) You can use the memory behind the Kernal ROM ($E000 to $FFFF) and the remaining memory between the ROMs ($C000 to $C3FF) for sprites

Machine code users don't have such a tough time since they aren't constrained by where they have to put their programs It is, nevertheless, useful to reconfigure the machine

Graphics Modes

Before we launch forth into graphics handling we must consider the graphics modes available to us The screen occupies 1000 bytes and is divided into 64000 addressable points or pixels There are two graphics modes allowing manipulation of the screen

1. Character Mode

In this default mode the screen uses 1000 characters each occupying an 8×8 pixel cell

2 Bit mapped Mode

In this mode, the screen uses a 320 by 200 array of pixels Using this mode it is possible to create pictures and other images

The fundamental difference between these modes is that character mode is supported by the operating system whereas bit map mode has no software to drive it Both modes use 8×8 cells to control the colours used

In addition to the graphics modes there are three colour modes

1 High resolution Mode

This is the default graphics mode In this mode, any given cell may contain only two colours the background or paper colour and the foreground or ink colour Any set pixel will have the ink colour and any unset pixel will have the paper colour

In character mode the paper colour is held in VIC register 53281 and the ink colour is held in the colour matrix

This mode allows the greatest detail, albeit at the most limited colour range

2 Multi-colour Mode

In this mode pairs of pixels are used to define dots of colour Since there are four possible arrangements for two pixels four colours are allowed in any given character cell (Table 2)

This mode is a lot coarser but allows greater colour flexibility

Extended Background

This mode uses high resolution but offers four different paper colours in addition to the usual ink colours The paper colour is determined by the POKE value of the character used and limits you to 64 different characters (Table 3)

Redefined characters

OK we've done the spade work let s now look at the use of user defined characters

You will have realised that the shape of characters is held in a table of data Exactly how is of essence Consider Figure 2 This shows a character design The design comprises of eight lines of dots Imagine that each set pixel is a 1 and each clear pixel is 0 That being so the top line becomes 00111100 The decimal equivalent of this binary number is 60 Similarly each line can be converted to a data value The character table comprises of a sequence of data values for each character The first eight data values in

Listing 2

```
10 DATA 60, 34, 34, 60, 34, 34, 60, 0
20 ch=4: FOR I=0TO7: READ X
30 POKE 51200+8*CH+I,X
40 NEXT
```

POKE CODE	COLOUR REGISTER
0-63	53281
64-127	53282
128-191	53283
192-255	53284

the table is used by the character normally used by @ The second block of eight is used by the character A And so on For any given character CH its data values start at

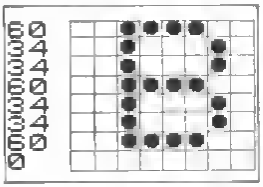
TABLE ADDRESS + CH*8

As an experiment, run Listing 1 as before putting the character table at 51200 Then type in and run Listing 2

Note what happens to the letter D

Using this approach is rather slow Listing 3 gives a machine code alternative

This code lives at $SC000 allowing you to use a relocated screen and character set as earlier The code has



two routines The first copies the ROM characters to a specified address, rather like lines 110 to 150 but faster You call the routine with the command

SYS 49152, ADDRESS

Listing 3

```
77  2000 FORL=OTO15 CX=0 FORD=OT
    O15 READA CX=CX+A POKE49152+
    L*16+D,A NEXTD
3A  2010 READA IFA<>CXTHENPRINT
    ERROR IN LINE",2040+(L*10) S
    TOP
DF  2020 NEXTL END
OD  2040 DATA76 6 152 76 153 192
     32 196 192 169 216 141 131
    3 169 208 2152
C4  2050 DATA141 130 3 169 0 133
    253 173 130 3 133,254 165,2
    0 133 251 2091
67  2060 DATA165 21 133,252 141
    232 3 141 233 3,173,14 220 4
    1 254 141 2167
2C  2070 DATA14 220 165 1 41 251
     133 1 160 0 177 253 145,251
     165 251 2228
A0  2080 DATA24 105 1 133 251 16
    5 252 105 0 133 252 24 165 2
    53 105 1 1389
B5  2090 DATA133,253 165 254,105
    ,0,133 254 165 253 208 222 1
    65 254 205 131 2500
4B  2100 DATA3,208,215,165 1,9 4
     133,1 102 21,173 14,220,9,1
    ,1279
0A  2110 DATA141 14,220 173 0 22
    1,41 3 160 173,232,3 56 249,
    145 192 2035
1C  2120 DATA141,232,3,78,232 3,
    78,232,3,173,24 208 41 240 1
    3 232 1533
C7  2130 DATA3 141 24 208 96 152
     128 64 0 32 227 152 32 206
    192 169 1506
DO  2140 DATAO 141 237 3 32 253,
    174 32,138 173 32 247 103 17
    2 237 3 2057
1F  2150 DATA165 20 145 251 200
    230,237 3,192 8 208 232,96 1
    73,238 3 2405
0E  2160 DATA141 239 3 96 32 253
     174 32 138 173,32 247,103 9
    6 6 251,2096
4F  2170 DATA38,252,6 251 38 252
    ,6,251,38 252,24 165,252 105
    ,233 3,2170
E4  2180 DATA133 252,96,32 196 1
    92 165,20 133 251 169 0 133,
    252 96,32 2152
34  2190 DATA144 97 32 134 97 16
    5,20,133,251 173,136 2,24 10
    5 4,133 1650
```

Where ADDRLSS is the start of the character table The routine also remembers where the character table is so always call it first in your program before trying to use the second routine

The second routine redesigns a specified character and has the form

SYS 49155 CH,B1 B2 B3 B4,B5,B6 B7,B8

Where CH is the character number and B1 to B8 are the bytes defining the character To redefine D as B as done earlier the command is

SYS 49155 4,60 34 34 60 34 34 60 0

For a bit of a laugh try this one

```
10SYS 49152 51200
20 FOR I=0TO999 POKE
50176+I 0 NEXT
30 FOR I=0TO7 A(I)=RND(I)*
256 NEXT
40 SYS 49155 0 A(0) A(1) A(2) A(3)
A(4) A(5) A(6) A(7)
50 GOTO 20
```

Listing 3 works equally well in multicolour and extended background modes To turn on multicolour character mode you must use

POKE 53270 PEEK(53270)OR16

To turn it off use

POKE 53270, PEEK(53270)AND239

To turn on extended background mode you must use

POKE 53265 PEEK(53265)OR64

To turn it off, use

POKE 53265 PEEK(53265)AND191

Bit Mapped Mode

This mode is a bit of a paradox Whilst on one hand it offers the greatest scope for artistic creativity it is also memory hungry In essence it uses two or three blocks of memory

1 The bit map

This is an area of 64000 pixels arranged in 200 lines of 320 This occupies 8K of RAM

2 The colour array

This holds the colour information and comprises of 1000 character cells

3 The color matrix

Multicolour mode uses this area to hold one of the colours -

The concepts discussed earlier with respect to telling the VIC chip where the bit map lies also apply here Register 53272 holds the details of the bit map area (bits 1 to 3) and the colour array (bits 4 to 7) In other words the bit map occupies the character set area and the colour array occupies the screen memory

To turn on the bit map you must turn on bit 5 of register 53265

POKE 53265 PEEK(53265)OR32

Register 53270 decides whether multicolour or high resolution modes is used

Multicolour on POKE 53270 PEEK(53270)OR16
Multicolour off POKE 53270 PEEK(53270)AND239

So how do we select the colours? In high resolution mode the colour array

holds this information. Each character cell value holds the paper colour in bits 0 to 3 and the ink colour in bits 4 to 7.

In multicolour colours 1 and 2 are held in the colour array with colour 1 in bits 4 to 7 and colour 2 in bits 0 to 3. Colour 0 is held in register 53281 and colour 3 is held in the colour matrix. The colours are displayed by the bit combinations

Colour 0	0 0
Colour 1	0 1
Colour 2	1 0
Colour 3	1 1

Since two pixels form a dot in multicolour mode, the horizontal resolution is limited to 160 points.

Any given pixel in the bit map is turned on with the following equations. Assuming that the pixel coordinates are X and Y and the bit map starts at the address BASE

```
ROW = INT(Y/8)
CHAR = INT(X/8)
LINE = Y AND 7)
BIT = 7-(X AND 7)
BYTE = BASE + ROW * 320 +
CHAR*8 + LINE
POKE BYTE, PEEK(BYTE) OR
2 ^ BIT
```

Using bit mapped mode from Basic is slow and over complex. Listing 4 gives a machine code package for handling bit mapped mode.

```
LISTING 4

EE  2000 FORL=01045 CX=0 FOR0=0T
    015 READA CX=CX+A POKE45152+
    L*16+D,A NEXTD
82  2010 READA:IFA<>CXTHENPRINT"
    ERROR IN LINE" 2040+(L*10) S
    TOP
0F  2020 NEXTL END
7E  2040 DATA76 26,192 76,170 19
    2 76 16 193 76,178 194,32 25
    3,174 32,1564
7A  2050 DATA138 173 32,247 183
    165,20 164 21 96 32,12,192 1
    41 136 3 1755
FC  2060 DATA32 12 192 141,132 3
    ,32,12 192 141,133 3 173,136
    3 240,1577
37  2070 DATA20 32 12 192 141 13
    4 3 32,12 192 141 135,3,173
    136 3 1361
9C  2080 DATA240 3,76,187,194 17
    3 132 3 10,10 10 10 13 133 3
    ,141 1338
```

```
E0  2090 DATA137 3 32 133 192,32
    218 192 173,2,221 3 3,141,2
    221 1711
CE  2100 DATA173,0,221,41,252,3
    0 141,0,221,173,17,208 9,32,
    141 1838
3E  2110 DATA17 208 169 121 141
    24 208,173 136,3 240 8,173,2
    2,208 9 1860
60  2120 DATA16 141 22 208,96 10
    9 220 133 169 169 0 133 160
    32 0,193 1869
D0  2130 DATA173 137 3 162 0 160
    ,127,145 160,136,16 251,72 2
    4,165,160 1515
E1  2140 DATA105,128 133 160 169
    0 101 169 133 169 104 202,2
    00,231 32,0 2060
C1  2150 DATA153,56 173,2 221,9
    3,141 2 221 173 0 221 41,252
    3,1752
29  2160 DATA3,141 0 221,173 17
    208 41,223,141,17,208 169,21
    ,141 24 1748
3F  2170 DATA208,173 22,208 41 2
    39,141 22 208 96 165,224 133
    ,169 169,0 2222
70  2180 DATA133 160 169 0 162,6
    4,160 127 145 160,136 16,251
    ,72 24 165 1560
D2  2190 DATA160 105,128 133,160
    169,0 101 169 133,169 104 2
    02 208 231 96 2284
5A  2200 DATA120 165 1,41,249 13
    3,1,96,165 1,9,6,133,1 88 96
    1305
A3  2210 DATA32,12,192 141 138 3
    ,140,139 3,32,12,192,141 147
    3,32,1359
FD  2220 DATA12,192,141,140,3,17
    3,136 3,240 3 76,235,193,32,
    77 193,1845
20  2230 DATA32 133 154 32,0 193
    ,173 147 3 160,0,174 140 3,2
    40,6 1630
55  2240 DATA32 166,154 76 73 19
    3,32,171,194,32,0 193 96 173
    ,147,3 1703
03  2250 DATA74,74 74,141 141 3,
    173,139 3,74,173 138 3 106,7
    4,74,1484
DB  2260 DATA141,142 3 173 147,3
    ,41,7 141 145 3 173 141 3 14
    1 143 1547
01  2270 DATA3,169,0 141 144 3,1
    62,6 32 220 193,202 208 250
    173 144 2050
27  2280 DATA3 133 171 173 143 3
    ,133 170 32 220 193 32 220 1
    93,24 173 2018
E0  2290 DATA143 3 101 170 133,1
    70,173 144 3 101,171,133,171
    169 0 141,1526
02  2300 DATA144 3,173 142 3 141
    ,143,3 32 220,193,32,220,193
    ,32 220 1894
90  2310 DATA193 24 173 143,3,10
    1 170,133 170 173,144 3,101,
    171,133,171 2006
48  2320 DATA24,173 145 3,101 17
    0 133 170 165,0 101,171 133
    171 24 169 1857
E9  2330 DATA0,101 170 133,170 1
    69,224 101,171 133,171 96 16
    9 0,14,144 1566
BC  2340 DATA3 14,143 3 105 144
    3,141 144,3 96 14,138,3,46,1
```

```
    38,143
B5  2350 DATA3 173 138 3 141,140
    3,173,139 3 141 143,3,173,1
    47,3 1540
54  2360 DATA141,150,3,173 140,3
    ,240,12,201 1,240,26,201,2 2
    40,33 1813
72  2370 DATA169,1,32 60,154,169
    ,1,32 60,154,36,169,0,32,60
    154,1463
FE  2380 DATA169,0,32,60,154,96
    169,0,32,60,154,169,1,32,60,
    154,1462
A0  2390 DATA96 169,1,32,60 154,
    169,0 32,60,154 96 141,151,3
    ,173,1571
C6  2400 DATA140 3,141,138,3,173
    143,3,141,139 3 173,150,3,1
    41,147 1655
F0  2410 DATA3,32,77,193 32,133,
    154 32 0,193 173,147 3 160,0
    ,174 1545
B5  2420 DATA151,3,240,6,32 166,
    154,76,109,194 32 171,194,32
    0 193 1783
89  2430 DATA24,173 140,3,105,1
    141 140,3,173,139,3,105,0,14
    1,139 1466
B6  2440 DATA3,32 0 193 96,173 1
    38,3,41,7 141 146,3,96,169 7
    ,1216
BC  2450 DATA237,146,3,141,146,3
    24 169 1 179,146 3 240 4 10
    202 1643
0E  2460 DATA208,252,141,147,3,8
    6,17,170 145 170,96,73,255,4
    9,170 145,2137
D6  2470 DATA170 96 32 12,192,14
    1 136,3,76,88,192,173,138,3
    141 33 1520
F1  2480 DATA200 173 136,3 141 1
    34 2 169 147,32 210,255 173
    133,3 10 1528
10  2490 DATA10,10 10,13,134,3 7
    6,79 192 151,0 151,0 191 0 1
    91,1231
```

This code starts at 49152 and has three routines

1 Activate bit map

This clears the bit map, set up the colours and turns on bit map mode. It has two forms

High resolution mode. SYS 49152,0,C1,C2

C1 = paper colour C2 = ink colour

Multicolour mode SYS 49152,1,C0,C1,C2,C3

C0 = paper colour

2 Return to text mode

This returns you to the normal text screen at its normal position, SYS 49155

3. Draw point

This draws the points at X,Y with the specified pen. SYS 49158,X,Y,PEN where

PEN = 0 draws the point in paper

colour, i e it erases the point
PEN = 1 draws the point in ink 1
PEN = 2 draws the point in ink 2
PEN = 3 draws the point in ink 3

In high resolution mode, X must be in the range 0 to 319 In multicolour mode X must be in the range 0 to 159 In either mode, Y must be in the range 0 to 199

In order to keep the routine as short as possible I have omitted any range checking of the co-ordinates If you use values outside the allowed range a crash may occur

4 Turn on bit map

Without clearing it SYS 49161, MODE MODE = 0, high resolution MODE = 1, multicolour

So that you don't loose any memory for Basic the bit map is placed behind the Kernal ROM and interface chip

Listing 5 is a simple demonstration

LISTING 5

```
A0  10 SA=12*4096
B4  20 SYS SA 1,15 11 12,0
B2  30 X1=5 Y1=5 X2=30 Y2=30 FA=
    1 GOSUB1000
B9  40 X1=10 Y1=10 X2=20 Y2=20 F
    A=2 GOSUB1000
13  50 X1=27 Y1=1 X2=35 Y2=50 FA
    =3 GOSUB1000
C6  60 FOR X = 0 TO 50
24  70 SYS SA+B,X+O,X,RND(1)*4 N
    EXT X
FA  80 O=O+1 IFO<10THEN60
DA  90 END
30  1000 FOR Y=Y1TOY2
78  1010 FOR X=X1TOX2
40  1020 SYS SA+B X Y,FA
45  1030 NEXT X Y
9A  1040 RETURN
```

Sprites

Sprites are probably the thing which makes games writers lives simplest To those of you who don't know, a sprite is a moveable block of 504 pixels arranged in a block of 21 rows of 24 The design is stored in a similar way to characters in that each row can be represented by 3 bytes with the whole design occupying 63 bytes These designs are stored as a sequence of blocks in any given bank The address of any given sprite block is given by

ADDRESS = (BANK*16384) + (BLOCK NO*64)

Specifying a sprite design

The next step is to tell the VIC which pattern block is to be used A maximum of eight sprites are possible and each has a pointer These pointers are located above the screen memory and can be found by

POINTER ADDRESS = SCREEN ADDRESS + 1014 + SPRITE NO

where SPRITE NO is from 0 to 7

A power up the screen sits at 1024 so the pointer for sprite 3 is at 1024+1014+3 or 2043 To tell the VIC which pattern to use you simply POKE the block number into the pointer eg to set sprite 1 to pattern 43

POKE 2041, 43

Turning on a Sprite

Whether or not a sprite is visible is determined by VIC register 53269 Each bit of this register controls a sprite To activate sprite SP use

POKE 53269 PEEK(53269) OR (2 SP)

To turn off sprite SN use

POKE 53269, PEEK(53269) AND 255-2 ^SP)

Expanded Sprites

Sprites can be expanded in both directions to give four possible sizes These are controlled by two registers To expand sprite SP in the X direction use

POKE 53277,PEEK(53277) OR (2 ^SP)

To reduce it again use

POKE 53277,PEEK(53277) AND (255-2 ^SP)

To expand sprite SP in the Y direction use

POKE 53271, PEEK(53271) OR (2 ^SP)

To reduce it again use

POKE 53271,PEEK(53271) AND (255-2 SP)

Colours

Each sprite has a colour register This is given by

REGISTER = 53287 + SPRITE NO

This is used to specify the colour of high resolution sprites

In multicolour sprites the colours are selected by the usual bit pairs see Table 3

The eight bits in register 53276 control the colour mode

To set a sprite SP to multicolour mode use

POKE 53276, PEEK(53276) OR (2 ^SP)

To set sprite SP to high resolution mode use

POKE 53276 PEEK(53276) AND (255-2 ^SP)

Positioning a Sprite

The positioning of any given sprite on the screen is defined by its X Y co-ordinates The X co-ordinate can range from 0 to 512 and Y co-ordinate from 0 to 256 Each sprite has a dedicated pair of registers The first holds part of the X position and the other holds the Y co-ordinates They can be found from

X Register = 53248 + SN*2

and the Y register is found from

Y Register = 53249 + SN*2

The X position is defined in two parts

Most significant byte (msb) = INT(XPOS/256)

Least significant byte (lsb) = XPOS-msb*256

Register 53264 holds the msb details one bit per sprite

So to position a sprite you use

POKE XREG,LSB
POKE YREG Y

If msb=1 then POKE 53264 PEEK(53264) OR 2 ^SP)
If msb=0 then POKE 53264 PEEK(53264) AND (255-2 ^SP)

Table 4

BIT PAIR	COLOUR SOURCE
0 0	Screen colour
0-1	Register 53285
1 0	Colour register
1 1	Register 53286

Selecting Colour Mode

Priorities

Each sprite has a priority which decides whether it appears in front of or behind the characters on the screen Register 53275 decides this one bit per sprite

To put sprite SP behind the characters use

POKE 53275, PEEK(53275) OR (2 SP)

To put sprite SP in front of the characters use

POKE 53275 PEEK(53275) AND (255-2 ^SP)

That's quite a mouthful and hardly conducive to simple programming Listing 6 gives the ubiquitous machine code package

This code has four routines

Setup Sprite

SYS 49408 SP TYPE COLOUR XEXP PRIORITY,[COLOUR1 COLOUR2)

where SP sprite number (0 to 7)
TYPE-0=High resolution 1=Multi colour
COLOUR - High resolution colour
XEXP - 1= X direction, 0=don't expand X direction
YEXP - 1=expand Y direction 0= don t expand y direction
PRIORITY-1=behind background 0=in front of background
COLOUR1 - Multicolour 1 only needed if TYPE =1
COLOUR2 - Multicolour 2, only needed if TYPE=1

Switch on

SYS 49411 SP FLAG where

FLAG=1 – turn on sprite SP
FLAG=0 – turn off sprite SP

Sprite position
SYS 49414 SP X Y where
SP – sprite number
X X position
Y – Y position

Pattern
SYS 49417 SP,DESIGN, BLOCK

The routine is quite smart in that it sorts out which bank you are using and where the sprite pointers are I therefore recommend that you use the configuration used earlier (screen at 50176 and characters at 51200 This allows you a block of 128 sprites from design block 128 to 255

In Summary

In all this has been a hefty slab of information and I must apologise for not giving more detail If you want to really get into graphics you must invest in the Programmers Reference Guide or something similar Having said that I believe that the routines I've given will be useful tools

```
62  2000 FORL=0TO30 CX=0 FORO=0T
    015 READA CX=CX+A POKE49408+
    L*16+O A NEXTO
82  2010 READA IFA<>CXTHENPRINT"
    ERROR IN LINE" 2040+(L*10) S
    TOP
0F  2020 NEXTL END
F1  2040 DATA76 194 194 76 12 19
    3 76 66 193 76,241,193 32 14
    4 194 32 1952
C3  2050 DATA134,194 165 20 41 1
    240 13 172 133 3 165 50,193
    13 21 1578
F0  2060 DATA208 141 21 208 96 1
    72 133 3 165 50,193 45,21 20
    8 141 21 1854
AD  2070 DATA208 96,1 2 4,8 16 3
    2 64 128 254 253 251 247 239
    223 2026
C3  2080 DATA191 127 32 144 194,
    32 134 194 165 20 164,21 141
    ,134 3 140 1836
3E  2090 DATA135 3,32 134,194 16
    5 20 141 136 3 173 133 3 24
    10 168 1474
83  2100 DATA173 134,3 153 0,208
    173 136 3 200 153 0 208 173
    135 3 1855
D1  2110 DATA240 13 172 133,3 18
    5 50 133 13 16 208,141 16,20
    8 96 172,1659
38  2120 DATA133 3 165 50,193,45
    ,16 208 141,16,208,96,41 1 2
    40,13 1597
F4  2130 DATA172,133,3,185 50 19
    3 13 28 208,141,28 208 96 17
    2 133 3 1766
68  2140 DATA185,50 193,45 28 20
    8 141,28 208 96,32,134 194 1
    65 20 41,1776
08  2150 DATA1 240 15 172 133 3
    185,50 193 13 28 208 141,28
    208 76 1636
57  2160 DATA206 193 172 133 3 1
    85 50 193 45 28 208,141 28 2
    08 32 134 1969
B0  2170 DATA194,165 20 41 1,240
    13 172 133 3 185,50 193 13
    28 208 1654
1F  2180 DATA141,28 208 96,172,1
    33,3 185 50,193 45 28 208 14
    1,28 208 1960
CA  2190 DATA96 32 144 194 32 13
    4,194 165 20 133 251 173 136
    ,2 24 105 1835
03  2200 DATA4 133,254 169,0 133
    ,251 56 165 253,233 0 133,25
    3 165 254 2466
F1  2210 DATA233 0,133 254 172,1
    33 3,165 251 145,253 96,192,
    128 64 0 2222
85  2220 DATA32 134 194 165 20 4
    1 1,240 13 172 133 3 185 50
    193 13 1589
54  2230 DATA27 208 141,27 208 9
    6,172 133,3 185 58 193 45 27
    208 141,1672
60  2240 DATA27 206 96 32,158 19
    4 169 0,141 142 3 32,95 194,
    172,142 1806
6C  2250 DATA3 145 253,200 192 6
    4 240,6 140 142,3 76,75 194,
    96 169 1558
15  2260 DATA255 141 14 212 141,
    15 212 169 128 141 16 212 16
    9 128 141 24 2120
86  2270 DATA212 173 27,212 96 3
    2 134 194 165 20 141 37 208
    32 134 194 2011
20  2280 DATA165 20 141 36 206 9
    6 32 253 174 32 138 173 32 2
    47 183 96 2026
96  2290 DATA32 134 194 165 20 2
    01 8 144,2 169 7 141 133 3,9
    6 32,1401
80  2300 DATA134 194 165 20 133
    253 169 0,133 254 162 8 6 25
    3 30 254 2174
50  2310 DATA202 208 248 173 0 2
    21 41 3 156 165 20 194 24 10
    1 254 133 2184
09  2320 DATA254 96 32 144 194 3
    2,134 194 165 20 141,141 3 3
    2,140 193 1515
7F  2330 DATA32 134 194 165 20 1
    72 133 3 153 39 206 32,170,1
    93 32 32 1712
52  2340 DATA194 173 141 3,240 3
    32 117 194 96 133 3 96 32 2
    28 255 1340
```

Swapper 64

Use part of your C64's memory as a RAM disk for storing Basic programs

By Chris Horton

Swapper 64 is a utility which allows a Basic program resident in memory to be stored in RAM for recall later rather like a RAM-disk. Commands are available for storing and recalling a program. Swapping the program in Basic memory for that in storage and storing part of a program (linenumber to linenumber).

The program can be used either in direct mode or in program mode: the latter being especially useful for utilising two programs held concurrently in memory – one in basic memory and one in storage. These can be rapidly interchanged when desired via the swap command.

Using the program

Upon initialising the program, the top of Basic memory is lowered to prevent a program overwriting the storage area. This leaves the user with 19K of Basic RAM which should be enough for most programs.
Swapper 64 has five commands which are activated by the syntax:

SYS 49152 command number (1-5),condition 1, condition 2

The commands are as follows:

1 – Initialise

SYS 49152,1

This lowers the top of Basic memory to $5400 to prevent overwriting the storage area. This should always be the first command executed when Swapper 64 is to be used.

2 – Store

SYS 49152,2

This command stores the resident Basic program into memory: this can be recalled using command three or swapped with another program using command four.

3 – Recall

SYS 49152,3 – recall program
SYS 49152,3,1 – recall program and auto-run

This function recalls a program in storage overwriting any program currently in basic memory. If a one is added to the SYS statement, the recalled program will auto-RUN. Make sure that there is a program in storage before you call this command.

Once recalled the storage area is NOT cleared. This means that a program can be recalled more than once if accidently NEWed.

4 – Swap

SYS 49152,4 – swap Basic program with stored program
SYS 49152,4,1 swap Basic program with stored program and auto-run

Using this command swaps the program in basic memory with a program stored in RAM. As in command three, if a one is added to the SYS instruction, the recalled program will auto-run. Again ensure that there is a program in stored memory before calling this instruction.

5 – Store part of a program

SYS 49152, beginning line, end line

This stores only the part of the program specified by the beginning line and end line numbers stated in the SYS statement. This commands main use is for when two programs are to be set up in memory to utilise the swap command. The programer can skip the initial lines of the first program stored (these are the lines that LOAD in the next part) so that they are not re-run every time this program is re-called from memory.
For example the first program LOAded would have initial lines something like those in Figure 1. The lines have the following function:

10 If swapper has been LOADed then skip lines 20 and 30

20 Set A to show that *SWAPPER* will have been LOADed

30 LOAD *SWAPPER* into memory

40 Initialise memory

50 SAVE the part of *PROG1* that is wanted

60 LOAD next program

Figure 1

```
10 IF A=1 THEN 40
20 A=1
30 LOAD "SWAPPER64",1 (or ',8' for disk) ,1
40 SYS 49152:1
50 SYS 49152:5,70,end line of program
60 LOAD "PROG2",1 (or ',8' for disk)
70 REM *START OF PROGRAM 1*
```

```
PROGRAM  SWAPPER LDR

10  10 BL=50:LN=70 SA=49152
F9  20 FOR L=0 TO BL CX=0 FOR D=
    0 TO 15
2F  30 READ A:IF A>255THENPRINT"
    NUMBER TO LARGE" LN+(L*10) S
    TOP
2B  40 CX=CX+A POKE SA+L*16+D A
    NEXT D
A9  50 READ A:IF A<>CX THENPRINT
    "ERROR IN LINE "(L*10) ST
    OP
4C  60 NEXT L:END
F1  70 DATA 32 253 174 32,138 17
    3 32 247 183,165 21 208,44 1
    65 20 240 2127
64  80 DATA 40 201,6 176 38 201
    1 240 23 201 2 240 13 201 3
    240 1824
64  90 DATA 12,201 4,240 14 201
    5 240 13 0 76 62 192 76 87 1
    92 1615
C4  100 DATA 76 71 193,76 80 193
     76 228 193 162 11 76,55 164
     32 110 1804
C0  110 DATA 192 160 0,177 251 1
    45 253 76 145 192 162-0 181
    43 157 59 2193
A4  120 DATA 195 232 224 14 208
    246 96 32 20 193 32 105 192
    32 110 192 2131
A6  130 DATA 160 0 177,253,145 2
    51 76 180 192,162 0 189 59 1
    95 149 43 2231
52  140 DATA 232 224 14,208 246,
    96 169 0 133 251 169 8 133,2
    52 165 45 2345
A1  150 DATA 141 39 195 165 46 1
    41 40 195 169 0 133 253 169,
    04 133 254 2157
AB  160 DATA 96 230 251 208 2 23
    0 252 230 253 208 2 230 254
    165 252 205 3088
03  170 DATA 40 195 240,3 76,67
    192 165 251 205 39 195 240 3
    ,76 67 2054
58  180 DATA 192 76,74 192 230 2
    51 208 2 230 252 230 253 208
     2 230 254 2884
37  190 DATA 165 252 205 40 195
    240 3 76 90 192 165 251 205
    39 195 240 2561
2F  200 DATA 3 76 90 192 165 43
    133 251,165,44,133,252,32,22
    ,195,160 1964
0B  210 DATA 0 169,0,145 251 76,
    59 193,230 251,208,2,230,252
    ,230 253 2549
BC  220 DATA 208,2,230,254,165 2
    52 201,84,240 3,76,137,193,3
    2,102,193 2372
E1  230 DATA 165,43,133,251,165
    44,133,252,32,22,195,160,0,1
    69,0 145 1909
22  240 DATA 251 76 59 193,169,4
    4,160,0 209,122 240 6,169,0
    144 37 1676
FA  250 DATA 195,96 32 253,174,3
    2,138 173,32 247 183,165 21
    208 237 165 2351
D1  260 DATA 20 201 1 208 231 16
    9 1 141 37 195,96 173 37 195
     208 1 1914
23  270 DATA 96 32 89 166 76 174
     167 169 255 133 55 169 83 1
    33,56,96 1949
FA  280 DATA 32 20 193 169 0 133
     251 169 8 133 252 169,0 133
     253 169 2084
0F  290 DATA 84 133,254 76 126 1
    93 162 0,181 43-141 41 195 1
    89 59,195 2072
B6  300 DATA 149 43 173 41 195 1
    57 59 195 232 224 14 208 235
     96 160 0 2181
B6  310 DATA 140 35 195 140 34 1
    95 140 38 195 177 251 141 41
     195 177 253 2345
81  320 DATA 145 251 201 0 208 1
    1,173 34 195 208 6 238 35 19
    5 76 164 2140
04  330 DATA 193 140 35 195 173
    41 195 145 253 173 35 195 20
    1 240 21 2238
48  340 DATA 165 252 197 46 208
    12 165 251 197 45 208 6 76 2
    15 193 76 2312
0D  350 DATA 253 192 76 232 192
    169 1 141,34 195 238 36 195
    173 36 195,2358
96  360 DATA 201 2 240 235 76 17
    6 193 238 36 195 173 36 195
    201 2 240 2439
53  370 DATA 222 76 194 193 32 2
    53 174 32 138 173 32 247,183
     165 20 141 2275
31  380 DATA 42 195 165,21 141,4
    3 195 32 253 174 32 138,173
    32 247 183 2066
98  390 DATA 165 20 141,44 195,1
    65 21 141,45 195,173,43 195
    205 45 195 1988
DD  400 DATA 240 25 176,34 165 4
    3 133 251,165 44 133,252,162
    0 181,43,2047
E5  410 DATA 157 59 195 232 224
    5 208,246,76,57 194,173 42 1
    95 205 44 2303
17  420 DATA 195 176 3 76 20,194
    ,76 57 192 173 42 195 141 48
    ,195 173 1856
04  430 DATA 43 195,141 40 195,3
    2 217 194 96 A9 00 8D 19
    5 32 22 1618
1F  440 DATA 195 165 251 141 38
    195 165 252 141 39 195 173 4
    6 195 133 251 2575
59  450 DATA 173 47 195 133 252
    173 44 195 141 48 195 173 45
    195 141 49 2199
E3  460 DATA 195 32 217 194 173
    46 195 133 251 173 47 195 13
    3 252 165 251 2652
BD  470 DATA 141 55 195 165 252
    141 56 195 177 251 141 57 19
    5 169 0 145 2335
9A  480 DATA 251 32 15 195 177 2
    51 141 58 195 169 0 145 251
    32 15 195 2122
28  490 DATA 173 38 195 133 43 1
    73 39 195 133 44 165 251 133
    45 165 252 2177
0A  500 DATA 133 46 32 62 192 16
    2 0 189 50 195 149 43 232 22
    4 5 208 1922
C4  510 DATA 246 173 55 195 133
    251 173 56 195 133 252 160 0
    173 57 195 2447
6E  520 DATA 145 251 200 173 58
    195 145 251 96 160 0 177 251
    141 46 195 2484
74  530 DATA 32 15 195 177 251 2
    40 24 141 47 195 32 15 195 1
    77 251 205 2192
E1  540 DATA 48 195 206 14 32 15
    195 177 251 205 49 195 208
    4 96 76 1960
BE  550 DATA 57 192 173 46 195,1
    33 251,173 47 195 133 252 76
    219 194 230 2566
B5  560 DATA 251 208 2 230 252,9
    6 166 251,202 134 251 224 25
    5 208 2,198 2930
56  570 DATA 252,96 0 0 0 0 0 0
    0 0 0 0 0 0 0 0 348
```


Continued on page 88

128 Disk Utility

Create auto boot disks and much more with this powerful utility

By K.R.Sharkey

How many times have you wanted to load a program from the directory and been frustrated to see the SYNTAX ERROR message appear when you press RETURN? The reason for this is that the letters PRG still existed on the command line when you pressed the RETURN key It is very annoying and is the main reason that I decided to write the C128 Disk Utility

If you wish to LOAD a program directly from the directory listing you must either remove the three characters (usually PRG) representing the filetype, from the line or you could place a colon before them before pressing return In effect your command line would look like this

DLOAD "program name" PRG

This is a valid command This program will modify any file you wish to enable you to simply type DLOAD before the file name in the directory listing It does this by modifying the disk directory and placing a colon outside the quotes but before the filetype flag so that each directory entry will look like the above

C128 Disk Utility has many other functions as well but the *put colon after filename* feature is probably the most useful It is well worth the trouble of obtaining a copy of *C128 Disk Utility* simply to have it available

Other Options

128 Disk Utility also offers the user the following functions

RENAME FILES
INITIALIZE THE DISK DRIVE
DISPLAY DIRECTORY OF DISK
MAKE A BOOT DISK
PRINT DIRECTORY IN DETAIL
DELETE FILES ON DISK
1541/1571 FORMAT A DISK
CLEAN UP A DISK
LOCK A FILE AGAINST SCRATCH
UNLOCK THE FILE AGAIN
PERFORM A QUICK SCAN AND DELETE
RECOVER A SCRATCHED FILE
QUIT THE PROGRAM
CHANGE SCREEN COLOURS

As you can see, *C128 Disk Utility* is a very valuable addition to your program library

Each function of the program is described below

RUNNING THE PROGRAM.

C128 Disk Utility is written completely in Basic V7.0, so it is simply LOADed SAVEed and RUN as any other Basic program There will be a short delay while the program initialises variables and strings

RENAMING FILES.

This is selected from the main menu as are the rest of the functions available

To select from the menu, move the pointer at the right of the menu block up and down using the cursor up/down keys, until it points to the function that you want

After you make your selection press RETURN to indicate to the computer that you want to accept the function the arrow is pointing to

Selecting RENAME FILES will gain entry to a sub-menu This menu works in exactly the same way as the main menu You may select from PLACE COLON AFTER FILENAME, RENAME FILES or GO TO MAIN MENU Couldn't be simpler right? Should you select the PLACE COLON AFTER FILENAME option you will be asked if you want to perform the operation on all the files on the disk or by selection from the directory Select the former and the machine will carry out its operation on all the files providing there is sufficient room after the filename for the necessary change to the directory

Any errors will be reported and the program always defaults back to the main menu after an error occurs

If you elect to choose files to be altered there will be a short delay while the program memorizes the directory and you will then be offered the files on the disk one after another and asked if you wish to alter them You will be able to select one of three possible replies to the prompt these are 'Y' for yes, 'N' for no, or "C for cancel everything and go back to the menu This holds true for virtually all modes in C128 Disk Utility

INITIALIZE DRIVE

This will clear the error channel on the drive and perform a dclear

DISPLAY DIRECTORY

Selecting this mode will print the directory on the screen You may slow the display down by pressing the COMMODORE key, or pause the display by pressing the NO scroll key Follow the prompt on the screen to return to the main menu

MAKE A BOOT DISK

As the name suggests, this allows you to make a disk which is BOOTable on the C128 computer A BOOTable disk is one which will load a nominated program, either by pressing the reset button, entering the command BOOT (RETURN) or switching the computer on with the disk in the drive Your C128 manual will explain this further To create a BOOT disk with C128 Disk Utility you should follow the procedure below

Select the disk you wish to BOOT It may have been used before or not If not it will have to be formatted first This is therefore the first question you will be asked, FORMAT DISK? (Y/N) Select

the appropriate answer See the part of this article pertaining to formatting disks for information about this

You now should have formatted in the drive either with or without any programs on it *C128 Disk Utility* will now perform a check of track one to ensure that there is no chance of writing over any data on the disk The program will report if it is safe to proceed, at the same time asking for your assurance that you want to continue If you reply NO there will be no harm done to the disk and you will be returned to the main menu however, if you wish to continue you will be asked for the name of the program to be BOOTed Upon answering this question you are asked if the program is a Basic program if you reply NO then the boot will be a non-relocatable boot otherwise the boot will be for a Basic program

The BOOT sector will now be written to the disk and you may SAVE the program you wish to boot if it is not already there

When you wish to LOAD and RUN the program simply BOOT the disk (see your Commodore manual for more details) and it will come up RUNning

PRINT DIRECTORY

This function prints out a detailed directory onto the printer or the screen To nominate the screen, switch the printer off and the program will default to the screen

Only one question needs to be answered in this mode and that is whether or not to show files in the directory which have been scratched This can be handy if you are looking for a deleted file on a disk which you may be able to recover using *C128 Disk Utility*

The printout of the directory will show the following information

DISK NAME
DISK ID
DISK FORMAT (usually 2A)
FILETYPE OF PROGRAM
TRACK WHERE IT LIVES
SECTOR ON THAT TRACK
FILENAME OF PROGRAM
NUMBER OF BLOCKS USED
START LOCATION IN C128
BLOCKS FREE ON DISK

DELETE FILES

This mode works exactly the same as **RENAME** except it deletes the file rather than renaming it You are still given the option of which programs to scratch from the disk If you make a mistake don't panic *C128 Disk Utility* can recover the program, so read on

FORMAT A DISK

Any disk to be used on a Commodore C128 must first be formatted Most of you will have learned this already but if not please refer to the manual which came with your computer for details

This section of the program allows you to format a disk without the need to know the necessary syntax All you have to know is whether you want the disk in 1571 double sided format or 1541 single sided format and what name you would like to give the disk

C128 Disk Utility takes care of things from now on If you wish you can even forget about the disk name and ID as the program will give the disk one for you Any errors occurring during the preparation of the new disk will be reported and eventually you will be passed back to the main menu

If you elect not to give a name and ID for the disk and the disk has been used before, the program will do a fast directory clear out on the disk which only takes a few seconds Please note that you cannot recover any programs from the disk with *C128 Disk Utility* once a format has been carried out, so be careful

COLLECT

Collect is the same as the COLLECT command in Basic V7.0. Disks in use should be COLLECTed periodically to ensure that they have a valid BAM (Block Allocation Map) and that they are using up disk space efficiently A COLLECT should always be done after you recover any scratched file using this program

LOCK/UNLOCK A FILE

It is not commonly known that it is possible to put a security lock flag on a file to prevent that file from inadvertantly being scratched This function does just that When you select this part of the program you will be passed to the FILETYPE EDITOR part of *C128 Disk Utility* Simply follow the prompts on the screen and you won t go wrong

If you elect to place a lock on a program it will appear in the directory as a'(symbol beside the filetype

By selecting unlock (simply press U when you are offered the program you wish to unlock) you can remove the lock

QUICK DELETE/RECOVER FILES

Once a program or file has been scratched from the disk it is still possible to recover it providing the area on the disk which the program occupies has not been over written by a subsequent write to the disk If it is at all possible to recover the file this feature of *C128 Disk Utility* will do it Select R when the program you wish to recover appears on the screen

You will now be asked which type of file to recover and your options are

0=DELETED (not normally used)
1=SEQUENTIAL (usually text)
2=PROGRAM (normally this one)
3=USER (special user design)
4=RELATIVE (usually a database)

Select the number corresponding to the filetype you require and the program will be recovered as that type of file It is always wise to carry out a COLLECT of the disk after this is done

Quick Delete is a much faster way to delete a selected file than the previous Delete function This is because it doesn t have to read the whole directory before it commences, it also does not clean up the BAM after it is finished

Should you press the return key at the prompt in the Filetype Editor then no change will occur on the disk and you will be offered the next file in the directory

QUIT

Quit does just that It restores the default Commodore colours and returns control to BASIC

CHANGE COLOURS

This has been added to the program just in case you don't like my colour selection. Enter a number between one and sixteen and you will be asked if it is correct. Answer YES or NO and press RETURN. If you selected yes the colour will be transfered to the screen and you will then be asked to select a new border and then a new character colour.

If you answer NO when asked if your selection is correct you will be asked again to enter a new colour.

I hope you find the *C128 Disk Utility* useful in maintaining your disk files.

```
PROGRAM  120 DISK UTIL


1 REM  ************************
**************************
***************
2 REM  *                      DI
SK UTILITY PROGRAM
               *
3 REM  *-----------------------
-----------------------
-----------*
4 REM  *
       BY
               *
5 REM  *
  K  B  SHARKEY
               *
6 REM  *-----------------------
-----------------------
-----------*
7 REM  *   TO USE SIMPLY RUN THI
S PROGRAM - IT WILL GIVE THE OPT
ION OF      *
8 REM  *   EITHER CONVERTING ALL
 FILES ON DISK OR SELECTING CERT
AIN FILES     *
9 REM  *   THE FILES WILL THEN BE
 RENAMED TO ALLOW OLOAD TO BE US
ED IN         *
10 REM *   FRONT OF THEIR NAME O
N THE DIRECTORY LISTING WITHOUT
THE           *
11 REM *   ANNOYANCE OF HAVING T
O INSERT A COLON AFTER THE FILE
NAME OR       *
12 REM *   CLEARING OUT THE REST
 OF THE LINE ON THE SCREEN
               *
13 REM *
               *
14 REM *  A BOOT DISK CAN BE CRE
ATED BY SIMPLY SELECTING  MAKE B
OOT DISK      *
15 REM *  ON THE MENU AND THEN A
NSWERING THE PROMPTED QUESTIONS
               *
16 REM *
               *
17 REM *  PROGRAMS CAN BE RENAME
D OR DELETED ON THE DISK IN THE
DRIVE BY      *
18 REM *  SELECTING THE CORRECT
COMMAND FROM THE MENU S ANSWERIN
G THE PROMPTS *
19 REM *
               *
20 REM *  LOCK/UNLOCK WILL ALLOW
 YOU TO LOCK A FILE SO THAT IT C
ANNOT BE      *
21 REM *  SCRATCHED OR UNLOCK IT
 AGAIN AT WILL  BESTORE DECODERS
 SCRATCHED     *
22 REM *  FILES  NOTE  IT IS WIS
E TO VALIDATE THE BAM AFTER UNSC
RATCHING      *
23 REM *
               *
24 REM *  SELECT  PRINT DIRECTOR
Y  AND A DETAILED DIRECTORY WILL
 BE LISTED ON *
25 REM *  THE POINTER  COLLECT
 WILL CLEAN UP THE BAM ON THE DI
SK             *
26 REM ************************
**************************
***************
27 COLOR RGB(0) 1 COLOR4,1 COLOR
5 15 IFRGR(0)=STHENPRINTCHR$(27)
;"R"
28 CLR O1=SSF1$),NA$(120) NE$(12
8) BE$=CHR$(77)+CHR$(15)
29 TY$(0)="DELETED    " TY$(1)="S
EQUENTIAL" TY$(2)="PROGRAM    " T
Y$(3)="USER     " TY$(4)="RELAT
IVE  "
30 TRAP282 REM ROUTINE TO HANDLE
 ERRORS
31
32 REM  **** DISPLAY MENU ****
33
34 SCNCLR FAST GS=1 YU=18 RESTOR
E38B IF M10$(SS(49),10,1 )="B"THE
N3S ELSEGOSUB230
35 IFROR(0)=STHENW1NDOW# 0 79,24
 ELSEWINDOW#,0 39,24
36 SLOW SCNCLR CHAR1 8 8 "     M
AIN MENU      ",1 GOSUB37 GOTO41
37 FORKT=4TOYU CHAR1,0 XT$$(XT)
 1 NEXT
38 CHAR1 1,17,"USE CRSR U/D TO S
ELECT"
39 CHAR1 1,18,"PRESS RETURN TO A
CCEPT SELECTION"
40 X=30 Y=5 CHA81,X,Y,"[RVSON] [
RVSOFF]" RETURN
41 GOSUB54
42
43 IF GSTHEN ON Y GOSUB,,,40 80,
85 S1,132 137 142,171,305 309 17
5 372
44 IFGS=8THEN ON Y GOSUB,,,S1,75
,78 RETURN
45 RETURN
46
47 REM  **** RENAME FILES ****
48
49 GOSUB 187 SCNCLR
50 FAST GS=0 YU=8 RESTORE 368 GO
SUB230 SLOW
51 CHAR1 5,0 "     RENAME MENU
    " 1
52 GOTO35
53 REM  ****  MAIN LOOP  ****
54 DO GETA$ LOOP UNTIL A$= [DOWN
] OR A$="[UP]" OR A$=CHR$(13)
55 IFA$="[DOWN] "THENPS=  " GOSUB
58 Y=Y+1 PS= [RVSON] [RVSOFF]
GOSUB55
57 IFA$="[UP]"1HENPS=    GOSUB55
 Y=Y-1 PS="[RVSON] [RVSOFF]" GOS
UB55
58 GOTO54    STOP
59 IFY>YUTHEN Y=4 ELSE IF Y<4THE
N Y=YU
60 CHAR 1 X Y PS  CHAR$ 0 Y SB(Y
) CHAR1 0 Y,SS(Y) 1   RETURN
61 PRINT"[CLEAR]RENAME USING COL
ON" PRINT"[DOWN][DOWN]APPLY COLO
N TO ALL NAMES [RVSON](Y/N/C)[RV
SOFF]"
62 DO GETA$ LOOP UNTIL A$="N" OR
 A$="Y" ORA$="C"
63 IF A$="Y THEN 70
64 IF A$="C THENPRINT"OK -CANCEL
ED" SLEEP3 GOTO48
65
66 REM  **** SELECTED NAMES ****
*
67
68 GOSUB 183
69 GOSUB215 SC=0 SCS= WRITE COLO
N AFTER  GOSUB 245 CLOSE1 RETURN
70
71 REM **** RENAME ALL ****
72
73 GOSUB218 GOSUB 258 CLOSE1 RET
URN
74
75 REM **** RENAME FILES ****
76
77 GOSUB218 SC=0 SCS="CHANGE  OD
SUB271
78 CLOSE1 RETURN
79
80 REM  ****  INITIALIZE ****
81
82 UTS="" PRINT"[CLEAR]  SS(5) C
LOSE15 OPEN15 8 15  10 " CLOSE15
  IFOSTHENGOSUB187 SLEEP4 :PPINT
83 RETURN
84
85 REM  **** DISPLAY DIR ****
86
87 PRINT"[CLEAR]" SS(0) CHAR1 1
1, "[RVSON]NO SCROLL[RVSOFF] KEY
TO PAUSE[HOME][DOWN][DOWN][DOWND
"
88 GOSUB183 DIRECTOPY 1FOSTHENPP
1N1CHR$(7) CHR$(15) DS$ SLEEP4 E
LSEPRINTCHR$(15),CHR$(7),"[DOWN]
PRESS A KEY  GETKEY A$
89 RETURN
90
91 REM  **** SETUP BOOT ****
92
93 PRINT"[CLEAR]  SS(7) NS=CHR$(
8) GOSUB183
94 CHAR1,8 3 "FORMAT DISK [RVSON
](Y/N)?[RVSOFF]"
95 DO GETA$ LOOP UNTILA$="Y OBAS
="N"
96 IFA$="Y"THENFLS=  " GOSUB191 R
```

```
EM GO DO FORMAT
97 OPEN15,8,15:OPEN8 8,8,"#" PRI
NT#15,"B-A ";0;1;0
98 IF DS=0 THEN PRINT PRINT"BOOT
 SECTOR IS FREE" FR=1 GOTO105
99 FR=0 PRINT#15,"U1";8;0;1;0
100 PRINT#15,"B-P";8;0 GET#8,C$
:IFC$=""CHR$(0)"ANDS$=""THENDS=
THENBEGIN PRINT#15,"B-F";8;7 PR
INT:RUS=""
101 DO GET#8,AS RUS=RUS+AS LOOP
UNTIL AS=""
102 IFRUS>" "THENRUS=" BOOT FOR "
RUS+ EXISIS
103 PRINTRUS BEND
104 PRINT"BOOT SECTOR IS IN USE"
105 PRINT PRINTCHR$(15),CHR$(7),
 CONTINUE OPERATION (RVSON)(Y/N)
(RVSOFF)
106 DO GETA$ LOOP UNTILA$="Y"ORA
$="N"
107 IFAS="N"ANDFR=1THENP#INT#15,
"B-F 8 1,0
108 IFAS= N"THENCLOSE8 CLOSE15 R
ETURN
105 DO
110 SCNCLR:CHART,8,24,"ENTER QUI
T TO RETURN TO MENU" 1
111 INPUT"(HOME)(DOWN)3(DOWN)(DOW
N)(DOWN)(DOWN)NAME OF BOOT PROGR
AM ,NNS
112 IFLEN(NN$)>10THENPRINTERR$(8
3) NN$=" SLEEP3 GOTO115
113 IFNN$=" "THENPRINTERR$(8) SLE
EP3
114 IFNN$="QUIT"THENPRINT"ABORTI
NG" SLEEP3 CLOSE8 CLOSE15 RETURN
115 LOOPWHILE LEN(NN$)>10ORNN$="
 "
116 PRINT"10 NWS PRINT"A BASIC
PROGRAM (RVSON1(Y/N)(RVSOFF)"
117 DO GETA$ LOOP UNTILA$="Y"ORA
$="N"
110 IFA$= N"THENBEGIN
119 PRINT"BOOT IS FOR NON RELOCA
TABLE PROGRAM"
120 MOS="BOOT BEND ELSEMOS="RU
N"
121 VB=LEN(NN$)+2831 VL=(VB/256
5) VH=INT(VB/256)
122 PRINT#15,"U1 8 0 T 0"
123 PRINT#15,"B-P 8 0
124 PRINT#8,"COM";NN$ N$ N$ N$ NN
$ N$ N$
125 PRINT#8 CHR$(162) CH0$(VL) C
HR$(1&0) CHR$(VH) CHR$(78) CHR$(
105) CHR$(175),
120 PRINT#8,MO$ CHR$(34) NNS,N$
127 PRINT#15,"U2 8 0 1 0"
128 CLOSE8 CLOSE15
129 PRINT"FINISHED " P0INT"
SAVE "NNS" TO THIS DISK " PRIN
T"AND IT WILL AUTOBOOT"
130 SLEEP5 RETURN
131
132 REM **** PRINT DIRECTORY **
**
133
134 PRINT"(CLEAR)" S$(8) PRINT"W
ILL 1 PRINT (RVSON)DELETED(RVSOF
F) FILES?" DO GETOAS LOOP UNTIL
OA$= Y OROA$="N"
135 GOSUB103 CLOSE2 OU=1 GOSU810
9 CLOSE1 CLOSE2 RETUSN
136
137 REM **** DELETE FILES ****
138
139 GOSUB219 SC=1 SC$="DELETE" G
OSUB 271
140 CLOSE1 RETURN
141
142 REM **** FORMAT DISK ****
143
144 PRINT"(CLEAR)",S0(10) IDS=HE
XS(INT(RND(0)*65535)) IDS=MIDS(I
DS,2,2) DCLEAR DIS="" SES=" " I
FOS =@TKENBEGIN
145 PRINT"(HOME)(DOWN)",CHR$(27)
,"@THIS DISK HAS BEEN FORMATTED
BEFORE"
146 PRINT"(HOME)(DOWN)(DOWN)",CH
R$(27)"@CONTINUE ((RVSON)Y/N(RVS
OFF))" 1OR=""
147 DO GETA$ LOOP UNTILA$="Y"ORA
$="N"
148 IFA$="Y"ISENSES=" (RVSON)OLD
 DO(RVSOFF)" GOTO150
149 PRINT0ES,"(HOME)(DOWN)",CHR$
(27) "@FORMAT CANCELLED" SLEEP3
GOTO29
150 BEND
TST 1NPUT"FORMAT IN 154I OR 1571
 MODE TS7I(CLEFT)(CLEFT)(CLEFT)CLE
FT)(CLEFT)(CLEFT)1" DM
TS2 IFDM=T54ITHENM$="U0:"M0
153 IFDM=I97ITHENM$="U0:"MI"
154 PRINT"(UP)"CHR$(27)"0" INPUT
"INPUT NEW OI5K NAME DI5K@(CLEFT
)(CLEFT)(CLEFT)(CLEFT)(CLEFT)DC
LEFT1",N*$
155 PRINT"(HOME)(DOWN)",CHR$(27)
,"@PRESS RETURN FOR QUICK DIRECT
ORY" PRINT"CLEARQUT OR ENTER NEW
 10 FOR"
TS6 DO INPUT"FULL FORMAT ON DISK
",D1$
157 LOOP UNTIL LEN(D1@)<=20R ASC
(D1$)=@
TS8 IFD1$=CHR$(T3)TSENTS@
159 IFD1$()""THENIOR=D1$ 9ES=" "
 "
180 PRINT"(HOME)(DOWN)" CHR$(27)
,"@FORMATTING -"NM$ SES ID$
181 OPEN15,8,15 PRINT#15,M$ PRIN
T#15,"N0 "+NM$+","+IOR
182 PRINT#15 "U0)M1" CLORE15
183 PRINT"(HOME)(DOWN)",CHR$(271
,"@DISK IS READY FOR USE"
184 IFOR<>@THENBEGIN PRINT"(HOME
)(DOWN)",CHR$(27)"@DISK ERROR OC
CURRED "
185 PRINTCHR$(T5),CHR$(7),OR$
186 PRINT"TRY AGAIN ((RVSON)Y/N(
RVSOFF)"
187 DO GETA$ LOOP UNTILA$="Y"ORA
$="N" IFA$="N"THENPRINT"ABORTING
 OPERATION"
188 IFA$="Y"THENPRINT"OK" BEND O
DTO150
189 SLEEP3 RETURN
170
171 @EM **** VALIDATE DISK ****
172 PSINTCSR$(7), SCNCL@ P@INT
S$(11)
173 PRINTCHR$(15),"(DOWN)COLLECT
ING A DISK CAN DESTROY"
174 PRINTCHR$(15),"DIRECT ACCESS
 FILES - " PRINT" CONTINUE ((RVSO
N)(Y/N)(RVSOFF)"
175 DO GET AS LOOP UNTILA$="Y"OR
 AS="N"
176 IF AS="Y"THENPRINT"(UP)(UP)(
UP)" CHR$(27)"@OK VALIDATING D
ISK" COLLECT FLS="" DOSUS107 SLE
EP3:DCLEAR RETURN
177 PRINT"(UP)(UP)(UP)(UP)" CHR$
(27),"@OK - CANCELLED" SLEEP3 RE
TURN
178
179 REM **** END PROGRAM ****
180
181 SCNCLR PRINT"END PROGRAM -
ARE YOU SURE?" DO GETA$ LOOP UNT
IL AS="Y"ORA$="N"
182 IFA$="Y"TKENCOLOR0,12 COLOR
4 14 COLORS 14 CLR GRAPHIC RGR(
0),1 GRAPHICCLR NEW SYS14384 ELS
E RETURN
183
184 REM **** DCLEAR & CHECK ERRO
RS
185
186 DCLEAR
187 PES=BE$+" DISK ERROR - "+DS$
 IF DS THENCHAR1 0 24 PES I SLEEP
3 PRINT"(CLEAR1 GOTO29
188 RETURN
189
190 SEM **** GET DIRECTORY ****
191
192 FR$=" OPEN2,DV PRINT#2 DI
SK NAME - " IF =125 AND ST TH
EN PES=SES+ERR$(5) CHAR1 0 24 PE
$ 1 SLEEP2 SCNCLR DV=3 CLOSE2 GO
TO192
193 OPEN15 0 15 OPEN5 0 5 0 IF
DSTHENPRINTCHR$(7) CHR$(15),DS$
SLEEP5 RETURN
194 PRINT#15 "U1" 5 0 18 0 PRINT
#15 "B-P 5 144
195 DO GET#5,AS PRINT#2 A$ LOOP
 UNTIL A$=CHR$(160) DO GET#5,A$
LOOP UNTILA$()CHR$(160)
198 PRINT#2 10 - A$ DO GET#5
,A$ PRINT#2 A$ LOOP UNTIL A$=CH
R$(10$)
197 PRINT#2 "FORMAT -" DO GET#5
 AS PRINT#2 A$ LOOP UNTIL A$=CH
R$(10$) PRINT#2 FOR1=1TO80 PRINT
#2 "(S *)" NEXT
198 PRINT#2 "FILETYPE TRACK SECT
OR" " NAME "BLOCK5
 LOCATION" FOR1=1TO85 PRINT#2
"(S *1" NEXT
199 TR=10 SE=1 FI=TR FS=SE
200 PRINT#15 "U1",5,0 TR SE
201 PRINT#15 8-P",5 $ GET#5 TR$
 SE$ TR=ASC(TR$) SE=ASC(SE$)
202 8=2 FORN=1TOR PRINT#15 S-P"
 5,8
203 GET#5 AS 2=ASC(A$) IFZ<>@AND
Z-192>@THENZ-Z-192 ELSE1FZ<>@THE
N2=Z-128
204 1FOA$="N"ANDZ=@THENGOTO212 E
LSE FR$=FR$-TY$(2)- "
205 GET#5 AS TK=ASC(A$) IFTK=@TH
EN274 ELSEIFTK>@THENFR$=FR$+STR$
(TK) ELSEFR$=FR$ +STR$(TK)
206 GET#5 AS SK=ASC(A$) IFSK>@TH
ENFR$=FR$+STR$(SK) ELSEFR$=FR$+
 "+STR$(SK)
207 FR$=FR$+" " FOR1=
3TO10 GET#5 AS FR$=FR$+AS NEXT1
208 FOR1=1STO20 GET#5 AS NEXT LO
-ASC(A$) GET#5 AS X1=FSC(A$) I=L
0+256*X1
209 IF1<>@THENFR$=FR$+" "+STR$(
1) ELSE1F1<>5THENFR$=FR$+ "+ST
R$(1) ELSEFR$=FR$+STR$(1)
210 IFZ<>@THENPRINT#15 "U"T",5,0
TK SK PRINT#15 "S-P" 5 2 GET#5 A
```

```
$ LO=ASC(A$) GET#S A$ H1=ASC(A$)
 Q=LO+256*H1 PRS=PRS+"
 "+STR$(Q)
211 PRINT#2 PRS
212 PRINT#15 "U1  S 0,PT 28 R-0+
32 PRS=   NEXTN
213 IFASC(TRS)<>0THENRT=TR PS=SE
 GOTO200
214 FORJ=1TO80 PRINT#2 "{S *}"
NEXJJ PRINT#2 CLOSE5
215 NUS=CHR$(0) OPEN1 8,0 "$0 $%
$ " GET#1 A$ A$ AR A$ AR A$
216 GET#1 A$ IFAR<>""THEN216
217 GET#1 A$ A$ AR 0$ BC=ASC(A$+
NU$)+ASC(R$+NU$)*256
218 PRINT#2 CHR$(27)" "CHR$(11)"@
** BC  "*** BLOCKS AVAILABLE "C
HR$(27)"- "CHR$(0) CLOSE1 CLOSE15
 SLOW 1FQU=3THENPRINT"PRESS A KE
Y  OO GETA$ LOOPWH1LEA$="" RETU0
N
215
220 REM **** GET NAMES ****
221
222 OPEN15 8,15,"I0" PRINT#15,"M
-R CHR$(11)CHR$(0)CHR$(12) UT$=""
223 FOR1=1TO2 GET#15 A$ UT$=UT$+
AR NEXT
224 1FUT$=FL$THENCLOSE15 RETURN
225 FL$=UT$ CLOSE15
226 FAS1 FORXT=0TO120 NA$1XT)=""
 NEXT
227 OPEN1,8,0 "$0" 1FOSTHENSLOW
GOSU@107 SLEEP1 RETURN
228 CO=0
229 SCNCLR GET#1,A$ A$
230 GET#1,A$ A$ IFAS=""THEN SLOW
 RETURN
231 GET#1 A$ 0$
232 GET#1 A$ IFAS=""THENGOTO238
233 1FAS=CHR$(134)THEN0E01N CO=CO
+1
234 AS= " 00 UNTIL A$=CHR$(34) G
ET#1 A$ 1FAS<>CHR$(34)THEN NA$(C
O)=NA$(CO)+A$
235 LOOP @END
236 GOTO232
237 SLOW RETURN
238
235 REM **** GET BYTES FROM D1SK
 ****
240
241 SCNCLR GET#1 A$ A$
242 GET#1 A$ A$ IFAS=""THENGOTO3
4
243 GET#1 A$ R$
244 SET#1 A$ IFAS= "THENGO]0242
245 RETURN
246
247 REM **** SELECT NAMES ****
248
249 SOSUB 322
250 FORXT=2TOCO PRINT CHOME]1DOW
N]CDOWN]CDOWN]CDOWN]

251 PRINT CHOME]CDOWN]CDOWN]CDOW
N]CDOWN]  CHR$(27)  @RENAME    "
NA$(XT)   CRVSON]CY N C]CRVSOFF]

252 DO GET A$ LOOP UNTILA$= Y"OR
A$= N"ORAN= C   1FAR= "C"THENPRINT
 OK   CANCELLED  SLEEP3 GOTO257
253 1FAS= Y THENSEGIN NE$(X)=LE
FT$1NA$(X)) 14)+CHR$(160)+    FL
$= "
254 RENAME 0@ (NA$(XT)) TO (NE$1
X))) U8   @END
255 1FOSTHENPRINTCHR$17),CHR$11S
) CHAR1 @ 24 DS$ 1 SLEEP2 GOSU$1
03 PRINT"1UP1 CHR$127) @"
256 NEXT
257 RETURN
258
258 REM **** CHANGE ALL NAMES **
 **
260
261 SCNCLR 1F CO=1THEN PRINTCHR$
17) CHR$11S)"NO FILES ON TH1S D1
SK" SLEEP3 RETURN
262 PRINT"CURRENT D1SK  -1RVSONJ
",NA$(1) "1RVSOFF] 1D  -1RVSONJ"
 UTR PRINT
263 PRINT"TH1S PROCEDURE WILL RE
NAME ALL FILES"
264 PRINT"CONT1NUE 1RVSONJ1Y/N]C
RVSOFF]" DO GET#S LOOPUNT1LA$="Y
"ORA$="N"
265 1FA$="N"THENRETURN
266 FL$="" FORXT=2TOCO PRINT"CHO
ME]1DOWN]CDOWN]CDOWN]CDOWN]CHANG
1NG  -"  NA$1XT)+"

267 NE$1XT)=LEFT$1NA$1XT) 14)+CH
R$11$0)+"
268 RENAME D@ 1NA$1XT)) TO 1NE$1
XT)) U@
260 1FOSTHENPRINTCHR$17) CHR$11S
) CHAR1 @ 24 DS$ 1 SLEEP1 PRINT"
CUP]"CHR$(27)"@" NEXT ELSE NEXT
270 RETURN
271
272 REM ****  @ENAME    ****
273
274 GOSU@302
275 PRINT PRINT FORXT=2TOCO
276 PRINT"CHOME]CDOWN]CDOWN]CDOW
N]CDOWN]CDOWN]"  9C$,    +  CHR$12
7)"@",NA@1XT) " CRVSON]1Y/N/C]CR
VSOFE)"
277 DO GET A$ LOOP UNT1LA@="Y"OR
A$="N"OR A$="C"
270 1FA$="Y"ANDSC=1THENSCRATCH 1
NA$1XT)) FL$="" GOTO284
275 1F A$="C"THENRR1NTCHR$115) C
HR$17),"OK - CANCELLED" GOTO2@@
280 1FA$="Y"ANDSC=@ THENBEGIN OU
$="" PRINT"CHOME]CDOWN]1DOWN]CDO
WN]CDOWN]CDOWN]",CHR$127)  @ENTE
R NEW NAME FOR " NA$1XT) 1NPU] O
U$
281 1FOU$=""THEN287
282 NE$(XT)=QU$
283 RENAME D@ 1NA$(XT)) TO 1NE$(
XT)),U@   ELS=""  @END
284 1FDSTHENBEGIN PRINTCHR$(7) C
XR$11S) CHAR1 @,24,DS$ 1
285 1FSC=@THEN CHAR1,@ 23,"BLANK
 NAME TO MOVE ON OR RE-ENTER NAM
E" SLEEP2
286 SLEEP2 DCLEAR PRINT"CUP]1UP]
CUP]"CHR$127)"@" SEND GOTO2@@
287 NEXT PRINTO5$ DCLEAR RETURN
288 SLEEP3 RETURN
289
290 REM  **** ERROR TRAP ****
291
292 FAST 1FREGR1@)-5THENW1NDOW@
@ 75,24 ELSEW1NDOW@ 5 39 24
293 1FER=5ANDDV=4THENDU=3 SLOW E
LSE1FER=5THENSLOW PES="@ES+ERR$1E
R) CHAR1,@ 24 PES 1 SLEEP3
294 ELS="" GOTO29
295
296 REM **** READ DATA FOR MENU
```

```
****
297
298 FORXT=4TOYU READ S$(XT)
299 OO S$(XT)=  "+S$(XT)+ "
300 LOOP UNT1L LEN1S$(XT))=>30
301 S$1XT)=LEFT$1S$(XT) 39) NEXJ
 RETURN
302
303 @EM ** PR1NJ OETA1LS ON SCRE
EN **
304
305 SCNCLR PRINT"CURRENT D1SK  -
(RVSON]"  NA$11)  1RVSOFF] 1D  -C
RVSON]",UT$ PRINT
306 1F CO=1THEN PRINTCHR$(7) CHR
$11S)"NO FILES ON TH1S D1SK" SLE
EP3 RETURN
307 PR1NJ  CHOME]1DOWN]CDOWN]SEL
ECT F1LENAMES TO   SC$
308 RETURN
309
310 REM ****      LOCK/UNLOCK
 ****
311 REM **** DELETE/RESTORE F1LE
S ****
312
313 GOSU@ 317 RETURN
314
315 REM ****   F1LETYPE EO1TOR
 ****
316
317 SCNCLR
318 OPEN15,8 15,"I" 1FOSTHENSCNC
LR GOTO106
315 OPENS,8 S  "  PR1NT"CCLEAR]C
R1GHT]CR1GHT]1R1GHT]CR1GHT]1 P1GH
T]CR1GHT]CR1GHT]CR1GHT]CR1GHT]CR
1GHT]CR1GHT1CR1GHT]F1LETYPE E01T
OR" TR=10 SE=1
320 PR1NT#15 "U1' S 0,TR SE PT=T
R PS=SE
321 PR1NT#15 "0-P",S 0 GET#S TR$
,SE$ TR=ASC(TR$) SE=ASC(SE$)
322 @=2 FORN=1TO0 PR1NJ#15 "@-R"
,S,@
323 GET#S,T$,DT@,O0$
324 T=ASC(T$) O@=ASC(O@@) DT=ASC
1UT$) 1FDT=@THEN335  REM LAST NA
ME NO TRACK
325 1FT>@ANDT-128<6THENT=T-128 L
$="          " GOTO327
326 1FT>@ANDT-152<6THENT=T-152 L
$="LOCKED  "
327 PR1NT"1HOME]CDOWN]CDOWN]CDOW
N]CDOWN]CDOWN] CHP$127) @FOUNO
-" PR1NT
328 FOR1=1TO18
329 GET#S AS 1FAS=""THENAS=CHR$(
@)
330 PR1NTAS,
331 NEXT1 PRINT"_{S *}CRVSONJ  L
$ TY$1T)   GOSU@330 @=@+32
332 1FKE$= "0"1XEN337
333 NEXT N
334 1FASC1TR$)<>@THEN32@ REM NOT
 LAST D1RECTORY SECTOR YET
335 CLOSE5   REM    CLOSE
336 CLOSE15  REM    EVERYTH1NG
337 RETURN
338
339 REM ****  GET ELECT1ON  ****
340
341 KE$= " PR1NT CHOME]CDOWN]COD
WN]1DOWN]CDOWN]CDOWN]1DOWN]CDOWN
]1DOWN]1DOWN]CRVSON]L1RVSOFF]-LO
CK 1RVSON]U1RVSOFF]-UNLOCK CRVSO
N]OCRVSOFF]-DELETE CRVSON]R1RVSO
```

```
EE)-RESTORE"
342 PRINT CRVSON)QERVSOFF)-QUIT"
  INPUT" [RVSON)RETURNERVSOFF) -
 NO CHANGE",KE$
343 IFKE$=""ORASC(KE$)<8  THENRE
TURN
344 IFKE$<>"L"ANDKE$<> U"ANDKE$<
> R"ANDKE$<>"O"ANDKE$<>"Q"THEN34
3
345 IFKE$="Q"THENCLOSE5 CLOSE15
RETURN
346 IFKE$="L"THENT-ASC(T$)OR64
347 IFKE$="U"THENT-ASC(T$)AND191
348 IFKE$="O THENT=0
349 IFKE$=" R'THENGOSUB355
350
351 REM ***WRITE MODECTION TO DI
SK***
352
353 PRINT#15,"B-P",5,0
354 PRINT#5,CHR$(T),
355 PRINT#15,"U2",5,0,PT,P5
356 IEKE$="R THENBEGIN PRINT"IT
 IS ADVISABLE TO COLLECT"
357 PRINT"THE DISK TO ENSURE A V
ALID BAM" BEND
358 FORXT-0TO50 NEXT RETURN
359
360 REM SELECT TYPE OF FILE
361
362 PRINT CHOME)CDOWN)CDOWN)CDOW
N)CDOWN)CDOWN)"CHR$(27)"0", FORX
T=0TO4 PRINTXT " -" TY$(XT) NEXT
XT
363 PRINT"SELECT FILETYPE TO RES
TORE" INPUTX
364 IFX>4ORX<0THENPRINT"CUP)" GO
TO363
365 T=X+128 RETURN
366 DATARENAME FILES,INITIALIZE
DRIVE DISPLAY DIRECTORY MAKE A B
OOT DISK PRINT DIRECTORY
367 DATADELETE FILES,FORMAT A DI
SK,COLLECT LOCK/UNLOCK FILE,DELE
TE/RECOVER FILES,QUIT SELECT COL
ORS
368 DATAPUT COLON AFTER F/NAME R
ENAME FILES,GO TO MAIN MENU
369 REM
370 REM SELECT SCREEN COLORS
371 REM
372 SCNCLR L1$=" [e J)[e *)[e *)[
[e *)[e *)[e *)[e *)[e *)[e *)[e
 *)[e *)[e *)[e *)[e *)[e *)[e *
)[e *)[e *)[e *)[e *)[e *)[e *)[
e *)[e *)[e *)[e *)[e *)[e *)[e
 *)[e *)[e *)[e *)[e *)[e *)[e *
)[e *)[e *)[e *)[e *)[e *)[e *)
[e *)[e I)"
373 L2$=" [e J)[e *)[e *)[e *)[e
 *)[e *)[e *)[e *)[e *)[e *)[e *
)[e *)[e *)[e *)[e *)[e *)[e *)
[e *)[e *)[e *)[e *)[e *)[e *)[
e *)[e *)[e *)[e *)[e *)[e *)[e
 *)[e *)[e *)[e *)[e *)[e *)[e *
)[e *)[e *)[e *)[e *)[e *)[e *)
[e *)[e *)[e *)[e *)[e *)[e *)[
K)"
374 SCNCLR PRINTL1@ PRINT " [e -
)       COLOR SELECTION MENU
    [e -)" PRINTL2@
375 WINDOW@ 12,39 23
376 SCNCLR PRINTL1@ PRINT " [e -
)       SELECT SCREEN COLOUR
    [e -)" PRINTL2@ INPUTS IE$<1
ORS>1@THENPRINTERR$(14) GOTO378
ELSECOLOR4,5
377 GOSUB350
378 IFA$<>"Y"THEN376
379 REM       FRGR(@)-STHENCOLOR
6,5 ELSECOLOR@,5
380 SCNCLR PRINTL1@ PRINT   [e -
)       SELECT BORDER COLOUR
    [e -)" PRINTL2@ INPUT@ IE$<I
ORB>1@THENPRINTERR$(14) GOTO380
ELSECOLOR4,@
381 GOSUB350
382 IFA@<>"Y"THEN300
383 REM  COLOR4,@
384 SCNCLR PRINTL1@ PRINT " [e -
)       SELECT PRINT COLOUR
    [e -)" PRINTL2@ INPUTFF IFFF
<1ORFF>1@THENPRINTERR$(14) GOTO3
84 ELSECOLOR4,FF
385 GOSUB350
386 IFA$<>"Y THEN384
387 REM   COLOR5,5
388 COLOR4,@ COLOR5,FF COLOR (RG
R(@)) 5 IFRGR(@)=STHENWINDOW@ @,
78,24 RETURN ELSEWINDOW@ @,30,24
 RETURN
389 A$=  INPUTA@ IFA@=""THEN389
 ELSE RETURN
390 SCNCLR PRINTL1@ PRINT " [e -
)       IS YOUR SELECTION OK?
    [e -)" PRINTL2@
391 GOSUB305
392 RETURN
```

LIFESAVERS	C64	BASIC PROTECTOR	1/1

These short Basic and machine code routines will enable a Basic program to be protected against prying eyes trying to LIST it.

They work by changing the LIST vector to point to a place in memory where a message is stored. The routine is situated at 53210 ($CFDA) and is called by SYS 53210.

To use the program to its full effectiveness, the routine should be incorporated in to a short auto-run loader program.

P.A.Eyes

```
1 DATA 169,229 160,207,141,5,3,1
42,3,96,169 239,160,207,32,30,
171,76 1,8
2 DATA 184,147,73,76,69,71,85,76
,32,69,78,84,82,89,46,46,46,00"
3 FORA=53210TO5324"READB POKEA,
B "E"T
4 S 553210
10 , program start address
20          *=$CFDA
30 , lo byte of new pointer
40          LDA #<NEWPNT
50 , hi byte of new pointer
60          LDY #>NEWPNT
70 , put into lo byte of LIST
vector
80          STA $0308
90 , put into hi byte of LIST
vector
100         STY $0309
110         RTS
120  lo byte of start of
message
130 NEWPNT  LDA #<TEXT
140 , hi byte of start of message
150         LDY #>TEXT
160 , print the message
170         JSR $AB1E
180 , back to Basic
190         JMP $A474
200 TEXT    BYT $93 $45,$4C,$45
,$45,$47,$41,$4C,$20,$45,$42,$59
,$52,$49,$22
210         BYT $2E,$2E,$00
220         END
```

New Characters on the MPS 801/3

Fed up with your MPS 801/3 character set? Now you can design your own characters

By Mathew Bittleston

When I bought my MPS 803 printer, I was very disappointed with the print quality – no true descenders and no chance of having Near Letter Quality (NLQ).

To overcome the most annoying problem of the two – no true descenders, I could think of two ways.

One – to print each line of text with two passes of the print head, the first to print the top part of the letters, the second to print the descenders.

Two – to re-define the character set so it's a little more squashed but has true descenders.

I chose the latter – partly because it would be quicker to print the text, and partly because I could design lots of different character sets more easily and they would take up less memory.

Presented here is that program – with a few extra frills. It is not very difficult to use – is as fast as it can be, because the printer is operating in bit image mode – and also uses up very little memory.

The character sets take up $0300 (768) bytes and can be stored anywhere in memory (except from $0000-$0800 (2048), obviously!) – including under the ROMs.

Getting it in

The program is presented here as a Basic listing called *PRINTER DRIVER*. Type this in using our *SYNTAX CHECKER* program that can be found with the *LISTINGS* article. Make sure that you SAVE the program to tape or disk before you RUN it.

If the program finds any errors when you RUN it it will indicate the line where the error has occurred. When you've got everything going O.K. type

```
SYS 52603
```

to initialise it, or

```
SYS 52600
```

to switch it off, which I doubt will be necessary. If you use RUN STOP/RESTORE you'll have to re-initialise it. Nothing should appear to have happened after initialisation. You can now use the printer as normal but everything will be printed through the driver.

There are a few more things that you may need to know.

OPEN X,Y,7 – will print in UPPER CASE/lower case as normal.

OPEN X,Y – will print in UPPER CASE only (shifted letters will be printed in lower case).

CHR$(27)CHR$(SET) – selects the character set where set is the most significant byte of the start address of the set i.e. If the start address of the character set is $FC00 (64512) the most significant byte is 64512/256=252.

CHR$(18) – will print in inverse characters until you de-select it with CHR$(146).

CHR$(8) – will print in Bit Image mode until you de-select it with CHR$(15).

CHR$(14) – will print in enhanced characters until you de-select it with CHR$(15).

All of the above CHR$ codes must be printed after a PRINT(No.)X instruction where X is the logical file number (see printer manual) and Y is the device number (normally 4).

Printing Rubbish

If you have typed in the Driver and initialised it don't be surprised when you get a whole load of garbage printed out. This is because you have to do more work, you have to type in a character set. There are five sets that I have designed included here – choose which you think will be the most useful, and type it in – or better still – type them all in. Again use the *SYNTAX CHECKER* to check your typing. The default character set that the driver software will print starts at 64512. If you wish to print from another set use

```
PRINT(No.)X CHR$(27)CHR$(start address/256)
```

The character sets can be relocated to any position in memory (I have positioned them under the KERNAL ROM) provided that the start address divided by 256 gives a whole number.

On your own

If you plan to design your own character sets you'll find the following information useful.

When designing the characters each number in the list of data stands for one

row which is printed The format in which the information is stored is shown in Figure 1

The important thing about the above is that it is different from when you define normal U D G s (for games etc) For the printer the rows are VERTICAL instead of horizontal

The MPS 803 can only print 7 dots vertically so in each byte there is one bit not used I've used this bit to tell the driver whether to print that row or not if bit 7 is SET then that row will be printed if it's NOT then it won't be This makes it possible to define character sets of different widths or even a proportional set The maximum width of a character is 8 rows, the normal MPS803 character width is just 6 rows

Now let me try to clarify what I've just said Look at Figure 1, from that you can see that because bit 7 is SET, row 0 will be printed In fact, if you look carefully at it you can see that nothing will be printed, but the print head will move one row to the right so that all the letters aren't jumed together Now look at row 7 bit 7 is NOT set so that line will not be sent to the printer

The line of data for the letter 'M' I have designed would be

DATA 128,191,130,140,191,0,0

NLQ?

I have thought of one way of producing a form of NLQ (???) on this printer

First print out your text remembering **exactly** how far up and across the paper was

Now put the paper back into the same position as last time, and print the text again If you do this just slightly wrong you'll get double images of every letter extremely annoying but do it right and you have a kind of NLQ on the MPS 803

Important Note

This program should work with most *Commodore* printers that will print in Bit Image Mode e g the MPS 801 but we haven't been able to test it with them all The program is not designed to work together with commercial software If you want to try it go ahead but, we can't guarantee what the results will be

Figure 1

Bit	Value	Row 0 1 2 3 4 5 6 7	
0	1	O * O O O * O O	This
1	2	O * * O * * O O	example
2	4	O * O * O * O O	shows
3	8	O * O * O * O O	'M
4	16	O * O O O * O O	from
5	32	O * O O O * O O	the
6	64	O O O O O O O O	Descender
7	128	* * * * * * O O	set
		128 130 130 0	
		191 140 191 0	

```
PROGRAM: DESCENDER.LDR

AE  100 REM * DRIVER CANNOT BE R
    ELOCATED *
88  110 SA=52500
A0  120 B=0 FORX=0TO11 READA:B=B
    +A POKESA+X,A NEXT SA=SA+X:R
    EADA:IFA=BTHEN120
06  130 IFA>0THENPRINT"DATA ERRO
    R IN LINE PEEK(63)+PEEK(64)*
    256
52  1000 DATA 76,51,205,169,0,14
    1,150,206,141,151,206,141,16
    37
43  1002 DATA 152,206,141,153,20
    6,141,155,206,160,205,169,64
    ,1958
7D  1004 DATA 140,39,3,141,38,3
    96,172,157,206,173,156,1324
65  1006 DATA 206,140,39,3,141,3
    8,3,96,234,141,140,206,1395
5A  1008 DATA 165,154,201,2,208,
    8,173,140,206,76,22,231,1563
17  1010 DATA 176,3,76,216,241,1
    73,140,206,72,32,65,205,1646
15  1012 DATA 76,243,205,173,152
    ,206,208,33,173,155,206,208,
    2038
FB  1014 DATA 46,173,148,206,201
    ,27,240,13,173,151,206,240,1
    824
72  1016 DATA 55,173,148,206,141
    ,154,206,166,0,141,151,206,1
    750
CE  1018 DATA 104,104,104,24,66,
    173,148,206,201,15,206,5,130
    8
C4  1020 DATA 166,0,141,152,206
    ,104,104,104,76,160,255,173,1
    652
61  1022 DATA 148,206,141,155,20
    8,240,225,169,32,32,243,205,
    2002
DA  1024 DATA 206,155,206,208,24
    6,76,128,205,173,148,206,201
    ,2158
4E  1026 DATA 8,208,8,141,152,20
    6,76,145,205,201,14,208,1570
84  1028 DATA 6,141,153,206,76,1
    28,205,201,15,206,11,166,151
    6
0F  1030 DATA 0,141,153,206,141
    ,152,206,76,128,205,201,10,16
    27
E0  1032 DATA 208,6,141,150,206
    ,76,128,205,201,146,208,8,168
    3
46  1034 DATA 169,0,141,150,206,
    76,128,205,201,16,208,8,1506
02  1036 DATA 141,155,206,76,128
    ,205,96,201,32,144,24,201,16
    09
12  1038 DATA 96,176,6,56,233,32
    ,76,24,206,201,192,144,1442
C7  1040 DATA 10,201,224,176,6,5
    6,233,128,76,24,206,169,1509
C1  1042 DATA 15,32,168,255,104
    ,76,168,255,133,251,165,185,1
    807
B0  1044 DATA 201,255,208,10,165
    ,251,201,32,144,4,73,66,1640
0C  1046 DATA 133,251,166,0,133
    ,252,6,251,36,252,6,251,1742
04  1048 DATA 36,252,6,251,36,25
    2,173,154,206,24,101,252,174
    7
10  1050 DATA 133,252,140,146,20
    6,142,148,206,166,8,32,168,1
    753
66  1052 DATA 255,160,7,120,32,1
    38,206,177,251,153,156,206,1
    864
A5  1054 DATA 136,16,248,32,138
    ,206,88,200,165,156,206,16,16
    30
32  1056 DATA 18,174,150,206,240
    ,2,73,127,174,153,206,240,17
    63
83  1058 DATA 3,32,168,255,32,16
    8,255,200,162,8,208,228,1746
AA  1060 DATA 172,146,206,174,14
    8,206,173,155,206,208,2,104,
    1603
83  1062 DATA 24,66,165,1,73,7,1
    33,1,96,0,0,0,566
DF  1064 DATA 0,0,0,0,0,0,0,252,0
    ,202,241,0,0,665
E1  1066 DATA 0,0,0,0,0,0,0,0,0,
    0,0,0,-1
```

PROGRAM: DESCENDER SET

```
70  100 REM * DESCENDER *
8C  110 SA=64512
42  120 PRINT"ESC CODE FOR SET ="
      CHR$(27)CHR$("SA/256")"
23  130 B=0 FORX=0TO11 READA B=B
    +A POKESA+X A NEXT SA=SA+X R
    EADA IFA=0THEN138
0A  140 IFA>0THENPRINT"DATA ERRO
    R IN LINE"PEEK(63)+PEEK(64)*
    256
50  1000 DATA 128,128,128,128,12
    8,128,0,0 128,128,128,222,13
    74
58  1002 DATA 128,128 0 0 128,13
    2,131,128,132,131 0 0 1030
48  1004 DATA 128,140,100,148,19
    0,148,0,0,128,132,170,255,16
    37
CD  1000 DATA 170,144 0 0,128,17
    7,137,184,183,128,0 0,1211
D3  1008 DATA 128,148,170,255,17
    0,182,0,0,128,128,128,132,15
    48
DC  1010 DATA 131,128,0,0,128,15
    0,182,183,128,128 0 0,1154
0A  1012 DATA 128,128,153,100,16
    6,128,0,0,128,170,156,100,16
    38
E1  1014 DATA 150,170,0 0,128,13
    0,138,128,136,138,0,0 1100
77  1010 DATA 128,128,128,152,17
    0,128,0,0 128,136,138,130,14
    10
E1  1010 DATA 138,138,0,0,128,12
    0,128,170,176,128,0,0 1130
55  1020 DATA 128,180,144,130,13
    2,130,0,0,128,100,177,173,14
    86
32  1022 DATA 183,158,0,0,128,12
    0,182,151,168,128,0,0 1210
10  1024 DATA 128,170,169,169,10
    5,186,0,0,128,145,181,161,16
    74
40  1020 DATA 166,155,0,0,128,13
    6,140,138,131,136,0 0 1189
7C  1028 DATA 128,151,165,106,16
    5,153,0,0,128,156,166,165,16
    42
C3  1030 DATA 165,152,0,0,128,12
    9,129,105,133,131,0 0 1152
05  1032 DATA 128,154,165,165,16
    5,154,0,0,128,134,169,169,15
    31
FF  1034 DATA 153,142,0 0,128,12
    0,128,164,164,128,0,0 1135
30  1030 DATA 128,128,128,196,10
    0,128,0 0 128,130,140,182,14
    62
C0  1030 DATA 162,128 0 0,128,14
    0,140,148,140,140 0 0,1150
10  1040 DATA 128,182,180,140,13
    5,128 0 0 128,130,128,169,14
    28
D6  1042 DATA 133,130,0 0 128,15
    0,161,173,169,134,0 0 1100
2D  1044 DATA 128,144,170,170,17
    0,180,0,0,128,151,164,162,16
    15
13  1046 DATA 162,156,0 0,128,15
    5,182,162,162,162 0 0 1250
AE  1048 DATA 128,150,162,162,16
    4,151,0 0,128,156,170,170,15
    87
E6  1050 DATA 170,148 0,0,128,12
    8,132,180,133,128 0,0,1150
D2  1052 DATA 128,148,210,210,21
    0,180,0,0,128,151,132,130,16
    85
00  1054 DATA 130,188 0,0,128,12
    8,164,185,180,128 0,0,1215
20  1056 DATA 128,128,152,157,10
    9,128,0,0,128,151,130,148,15
    57
10  1058 DATA 146,180,0,0 128,12
    0,161,151,180,128 0,0,1202
46  1060 DATA 128,150,130,180,13
    0,180,0,0,128,180,132,130,15
    34
30  1062 DATA 130,180 0 0,128,15
    0,180,180,162,156 0 0,1244
45  1064 DATA 128,254,146,180,16
    2,158,0,0,128,150,182,180,16
    16
01  1066 DATA 146,254,0,0,128,15
    0,132,130,130,132,0 0 1242
00  1000 DATA 128,164,170,170,17
    0,144,0,0,128,130,159,180,15
    25
00  1070 DATA 180,144,0,0,128,15
    0,180,168,180,180,0 0 1202
A7  1072 DATA 128,142,144,160,14
    4,142,0,0,128,158,160,152,14
    50
07  1074 DATA 160,158,0,0,128,18
    2,182,136,182,180,0,0 1270
E9  1076 DATA 128,142,200,200,20
    0,158,0,0,128,182,170,178,17
    22
60  1078 DATA 180,182,0,0,128,12
    0,255,153,153,128,0 0 1353
75  1060 DATA 128,180,158,185,10
    9,162,0,0,128,128,153,153,16
    28
42  1082 DATA 255,128,0,0,128,12
    8,138,151,138,128,0,0,1210
A5  1084 DATA 128,138,158,136,13
    6,138,0,0,138,136,136,138,13
    72
0E  1080 DATA 130,136,0 0 128,18
    0,138,137,138,180 0 0,1180
F0  1088 DATA 128,161,191,165,16
    5,154,0,0,128,158,161,161,15
    72
2C  1090 DATA 161,146,0,0,128,16
    1,151,161,146,148,0 0,1234
0F  1092 DATA 128,161,191,165,16
    5,161,0 0,128,161,151,165,16
    16
3E  1094 DATA 133,128,0,0,128,15
    0,161,161,169,185,0 0,1224
02  1090 DATA 128,151,132,132,13
    2,151,0,0,128,128,161,191,15
    14
9D  1090 DATA 161,128,0,0,128,14
    0,160,161,159,128,0 0,1170
6E  1100 DATA 128,128,151,148,14
    6,161,0,0,128,161,151,161,15
    35
59  1102 DATA 180,176,0,0,128,15
    1,138,148,138,151,0 0,1246
2C  1104 DATA 128,191,130,132,13
    6,151,0,0,128,158,161,161,15
    16
03  1100 DATA 161,158,0 0,128,16
    1,191,165,137,134,0 0,1239
4A  1108 DATA 128,158,161,165,14
    5,174,0,0,128,161,151,137,15
    52
30  1110 DATA 153,166,0,0,128,14
    5,165,165,165,152,0,0,1240
9C  1112 DATA 128,131,151,191,16
    1,131,0 0,128,159,168,180,15
    10
24  1114 DATA 160,159,0 0,128,13
    5,152,180,152,135,0 0,1101
D2  1116 DATA 128,159,160,156,16
    0,155,0,0,128,101,179,148,15
    36
48  1110 DATA 175,181,0,0,128,13
    1,164,180,164,131,0 0,1246
02  1120 DATA 128,183,177,173,16
    3,177,0 0,136,138,136,255,16
    44
20  1122 DATA 136,138,0,0,0 05,4
    2,05,0,0,0 0,1484
04  1124 DATA 0 0 0 0,127 0,0,0,
    0 4,00 4,185
00  1120 DATA 20 2,0,0,0,00 05,9
    3,05 05 02 0 -1
```

PROGRAM: MODERN SET

```
CE  100 REM * MODERN *
90  110 SA=63744
42  120 PRINT ESC CODE FOR SET =
      CHR$(27)CHR$( SA/256 )
23  130 B=0 FORX=0TO11 READA B=B
    +A POKESA+X A NEXT SA=SA+X R
    EADA IFA=0THEN130
D4  140 IFA>0THENPRINT DATA ERRO
    R IN LINE PEEK(63)+PEEK(64)*
    256
81  1000 DATA 128,128,128,128,12
    8,128,128 0 128,128,172,175,
    1499
44  1002 DATA 128,128,128,0 128,
    134,135,128,134,135,128 0 13
    D6
06  1004 DATA 128,138,186,191,13
    0,191,138 0,128,174,170,255,
    1837
55  1006 DATA 170,186,186 0,128,
    179,187,152,132,180,179 0,16
    79
06  1008 DATA 128,188,191,165,17
    3,144,168 0,128,128,128,133,
    1674
66  1010 DATA 131,128,128,0,128,
    188,250,225,193,128,128 0,16
    27
C5  1012 DATA 128,128,193,225,25
    0,188,128 0,128,170,156,190,
    1804
D3  1014 DATA 156,170,128 0,128,
    136,136,184,190,136,138 0,15
    00
60  1016 DATA 128,128,128,208,17
    6,128,128,0,128,136,136,136,
    1560
19  1018 DATA 136,136,128 0,128,
    128,128,176,176,128,128 0,13
    92
3C  1020 DATA 128,160,176,152,14
    0,130,129 0,128,184,191,169,
    1687
27  1022 DATA 165,163,191 0,128,
    128,184,191,128,128,128 0,15
    34
81  1024 DATA 128,184,185,169,16
    9,173,175 0,128,160,169,169,
```

```
    1809
72  1026 DATA 169 191 184 0 128
    143 140 136 184 190 136 0 16
    01
35  1028 DATA 128 175 173 169 16
    9 185 184 0 128 184 191 169
    1855
AF  1030 DATA 168 168 184 0 128
    129 129 165 185 133 131 D 15
    4D
D2  1032 DATA 128 184 191 169 16
    9 173 191 0 128 143 139 137
    1752
15  1034 DATA 185 191 128 D 128
    128 128 148 148 128 128 0 14
    40
90  1036 DATA 128 128 128 212 18
    0 128 128 0 128 128 152 10D
    1620
1F  1038 DATA 226 193 128 0 128
    148 148 140 148 140 128 0 15
    43
D4  1040 DATA 128 128 193 226 18
    0 152 128 0 128 131 129 217
    1740
8C  1042 DATA 217 139 143 0 129
    158 161 173 173 169 142 0 16
    04
62  1044 DATA 128 184 191 137 13
    7 139 191 D 128 191 185 169
    1780
05  1046 DATA 171 175 184 0 128
    184 191 161 161 163 179 0 16
    97
72  1048 DATA 128 184 191 161 16
    2 164 184 0 128 184 191 171
    1840
6F  1050 DATA 169 169 160 0 128
    184 191 139 137 137 128 D 15
    42
F9  1052 DATA 128 184 188 162 16
    1 161 185 D 128 184 191 136
    1808
9C  1054 DATA 136 140 191 D 128
    128 184 191 131 128 128 0 14
    85
A0  1056 DATA 128 184 184 160 19
    1 131 128 0 128 184 191 136
    1745
C9  1058 DATA 140 154 185 0 128
    184 191 160 160 176 176 0 16
    54
99  1060 DATA 128 184 191 129 14
    3 129 191 0 128 184 191 129
    1727
D7  1062 DATA 129 131 191 0 128
    184 191 161 161 163 191 0 16
    3D
FD  1064 DATA 128 184 191 137 13
    7 139 143 0 128 184 191 161
    1723
E5  1066 DATA 169 147 175 D 128
    184 191 137 153 171 143 0 15
    98
BE  1068 DATA 128 144 166 169 16
    9 187 187 0 128 129 185 191
    1783
87  1070 DATA 129 131 131 0 128
    184 191 160 160 163 191 0 15
    69
A1  1072 DATA 128 152 191 160 16
    0 147 143 0 128 184 191 16D
    1744
FC  1074 DATA 184 163 191 0 128
    176 187 140 140 139 176 D 16
    24
B5  1076 DATA 128 143 139 184 18
    4 140 143 D 128 160 177 165
    1711
FD  1078 DATA 169 165 163 D 128
    128 248 255 193 128 128 D 17
    05
B3  108D DATA 128 16D 184 191 16
    9 171 163 D 128 128 192 249
    1863
41  1082 DATA 255 128 128 0 128
    132 242 255 130 132 128 0 16
    58
80  1084 DATA 136 136 136 168 18
    6 156 136 0 D D 0 255 1309
D9  1086 DATA 255 0 0 255 128 18
    4 191 137 137 139 191 D 1617
9D  108B DATA 128 191 185 169 17
    1 175 184 D 128 184 191 161
    1867
1B  109D DATA 161 163 179 D 128
    184 191 161 162 164 184 0 16
    77
10  1092 DATA 128 184 191 171 16
    9 169 16D D 128 184 191 139
    1814
4C  1094 DATA 137 137 128 0 128
    184 188 162 161 161 185 D 15
    71
0A  1096 DATA 128 184 191 136 13
    6 140 191 0 128 128 184 191
    1737
B0  1098 DATA 131 128 128 D 128
    184 184 16D 191 131 128 0 14
    93
F6  11D0 DATA 128 184 191 136 14
    0 154 185 0 128 184 191 16D
    1701
84  1102 DATA 16D 176 176 D 128
    184 191 129 143 129 191 0 16
    07
E4  1104 DATA 128 184 191 129 12
    9 131 191 D 128 184 191 161
    1747
79  1106 DATA 161 163 191 D 128
    184 191 137 137 139 143 0 15
    74
AD  1108 DATA 128 184 191 161 16
    9 147 175 0 128 184 191 137
    1795
A7  111D DATA 153 171 143 D 128
    144 166 169 169 187 187 D 16
    17
A9  1112 DATA 128 129 185 191 12
    9 131 131 D 128 184 191 16D
    1607
01  1114 DATA 160 163 191 0 128
    152 191 160 160 147 143 0 15
    95
E0  1116 DATA 128 184 191 160 18
    4 163 191 0 128 176 187 140
    1832
07  1118 DATA 14D 139 176 0 128
    143 139 184 184 140 143 0 15
    16
C6  112D DATA 128 16D 177 185 16
    9 165 163 D 128 128 248 255
    1906
1C  1122 DATA 193 128 128 0 128
    160 184 191 169 171 163 D 16
    15
7E  1124 DATA 128 128 192 249 25
    5 128 128 0 128 132 242 255
    1965
E6  1126 DATA 13D 132 128 0 136
    136 136 168 186 156 136 D 1
```

PROGRAM ITALIC SET

```
6D  10D REM * ITALIC *
94  11D SA=62976
42  12D PRINT ESC CODE FOR SET -
    CHR$(27)CHR$( SA/256 )
23  130 B=0 FORX=0TO11 READA B=B
    +A POKESA+X A NEXT SA=SA+X R
    EADA IFA=BTHEN130
04  140 IFA>0THENPRINT DATA ERRO
    R IN LINE PEEK(63)+PEEK(64)*
    256
5D  100D DATA 128 128 128 128 12
    8 128 0 0 128 128 192 144 13
    60
E1  10D2 DATA 140 131 0 0 128 13
    3 131 128 133 131 0 0 1055
CC  1004 DATA 144 244 156 247 15
    6 151 0 0 128 128 164 234 17
    52
04  1006 DATA 171 145 0 0 128 17
    6 137 173 164 131 0 0 1226
6A  1008 DATA 128 144 172 234 17
    1 130 0 0 128 128 128 133 14
    96
3C  1010 DATA 131 128 0 0 128 12
    8 152 164 194 129 0 0 1154
A5  1012 DATA 128 128 192 161 14
    6 140 0 0 129 170 156 190 15
    39
E1  1D14 DATA 156 170 0 0 128 13
    6 136 190 136 136 0 0 1180
05  1016 DATA 128 128 128 208 17
    6 128 0 0 128 136 136 136 14
    32
E1  1018 DATA 136 136 0 0 128 12
    8 128 176 176 128 0 0 1136
80  1020 DATA 128 160 144 136 13
    2 13D 0 0 128 156 170 161 14
    45
0D  1D22 DATA 149 142 0 0 128 12
    8 160 186 167 128 0 0 1180
52  1024 DATA 128 176 170 169 16
    9 134 0 0 128 144 161 165 15
    44
D5  1026 DATA 155 129 0 0 128 13
    6 140 17D 157 139 0 0 1154
1C  1028 DATA 128 144 166 165 16
    5 153 D 0 128 156 166 165 15
    36
DC  1030 DATA 165 153 0 0 128 16
    0 145 137 133 131 0 0 1152
D8  1032 DATA 128 144 170 165 16
    5 154 D 0 128 166 169 169 15
    58
A3  1034 DATA 153 142 0 0 128 12
    8 144 148 132 128 0 0 1103
5E  1036 DATA 128 128 208 180 13
    2 128 0 0 128 136 148 162 14
    70
51  1D38 DATA 161 129 0 0 128 14
    4 148 148 148 132 0 0 1138
86  104D DATA 128 160 161 145 13
    8 132 0 0 128 128 194 153 14
    67
A2  1042 DATA 137 134 0 0 128 15
    8 161 173 173 169 142 0 1375

F5  1D44 DATA 128 184 164 162 14
    6 174 0 0 128 176 174 165 16
    01
A3  1046 DATA 164 152 0 0 128 15
    2 164 162 162 148 0 0 1232
```

```
9C  1048 DATA 128,156,162,162,15
    6,163,0,0,128,152,172,170,15
    4,9
55  1050 DATA 17D,132,0,0,128,19
    2,200,190,137,129,0,0,1278
9A  1052 DATA 128,204,210,210,17
    0,150,0,0,128,160,158,133,16
    5,8
15  1054 DATA 164,152,0,0,128,12
    8,180,172,165,129,0,0,1218
05  1056 DATA 128,176,192,180,14
    1,129,0,0,128,160,158,137,15
    2,9
D7  1058 DATA 180,162,0,0,128,12
    8,184,166,165,131,0,0,1244
1C  1060 DATA 128,188,130,140,16
    2,156,0,0,128,176,142,132,14
    8,2
4D  1062 DATA 178,140,0,0,128,18
    4,164,162,146,140,0,0,1242
C8  1064 DATA 128,248,150,162,16
    2,156,0,0,0,136,148,21D,1500

A6  1066 DATA 170,158,128,D,128,
    162,150,132,130,132,0,0,1306

50  1068 DATA 128,160,164,17D,17
    0,146,0,0,128,120,180,174,15
    4,8
4F  1070 DATA 165,144,0,0,128,10
    4,166,160,152,166,0,0,1265
4D  1072 DATA 128,152,166,160,14
    4,142,0,0,128,180,146,184,15
    3,8
53  1074 DATA 144,142,0,0,128,16
    0,150,136,180,130,0,0,1170
9A  1076 DATA 128,204,210,200,18
    8,130,0,0,128,160,178,170,16
    9,6
65  1078 DATA 166,146,0,0,128,22
    4,220,195,129,129,0,0,1337
78  1080 DATA 128,2D0,248,206,20
    1,192,0,0,128,192,192,225,19
    2,12
64  1082 DATA 157,131,D,0,128,12
    8,228,154,135,130,4,0,1195
47  1084 DATA 136,136,136,160,15
    4,140,0,0,136,136,136,136,14
    4,14
1A  1086 DATA 136,136,0,D,128,15
    6,162,161,157,163,0,0,1159
3E  1088 DATA 128,104,167,165,16
    5,154,0,0,128,156,162,161,15
    7,0
47  1090 DATA 161,146,0,0,128,18
    4,167,161,145,142,0,0,1234
78  1092 DATA 128,104,167,165,16
    5,129,0,0,128,184,135,133,15
    1,8
5D  1094 DATA 133,129,0,0,128,15
    6,162,225,169,154,0,0,1256
80  1096 DATA 128,104,135,132,18
    8,135,0,0,128,128,160,185,15
    0,3
2C  1098 DATA 167,129,0,0,128,15
    2,160,161,153,133,0,0,1185
9C  1100 DATA 128,184,143,148,16
    2,129,D,0,128,184,166,161,15
    3,3
11  1102 DATA 160,144,0,0,128,19
    0,129,142,161,158,0,0,1212
7A  1104 DATA 128,184,135,129,18
    5,134,0,D,128,156,162,161,15
    D,2
57  1106 DATA 145,142,D,0,128,18
    4,143,137,137,134,0,0,1150

02  1108 DATA 128,156,162,169,14
    5,174,0,0,128,184,143,153,15
    4,2
CE  1110 DATA 169,134,0,0,128,14
    4,162,165,165,153,0,0,1220
40  1112 DATA 128,129,185,135,12
    9,129,D,0,128,152,167,16D,14
    4,2
9F  1114 DATA 184,135,0,0,128,15
    2,167,16D,144,143,0,D,1213
CD  1116 DATA 128,158,161,152,16
    0,159,D,0,128,160,147,14D,14
    9,3
D6  1118 DATA 14D,179,128,0,128,
    102,201,200,190,129,D,0,1477

66  1120 DATA 128,128,162,177,173,16
    3,145,0,0,136,136,136,255,16
    11
84  1122 DATA 136,136,0,0,0,0,0,
    0,0,0,0,0,272
88  1124 DATA 0,D,0,0,D,0,0,0,0,
    0,D,0,0
C5  1126 DATA D,0,0,D,0,0,0,0,0,
    0,0,0,-1
4F  1128 DATA 128,128
```

PROGRAM GOTHIC SET

```
8D  1D0 REM * GOTHIC *
83  11D SA=61440
42  120 PRINT"ESC CODE FOR SET =
    CHR$(27)CHR$("SA/256")
23  130 S=0 FORX=0TO11 READA S=B
    *A POKESA+X A NEXT SA=SA+X:R
    EADA 1FA=BTHEN130
04  140 1FA>0THENPRINT DATA ERRO
    R IN LINE PEEK(63)+PEEK(64)*
    256
39  1000 DATA 128,128,128,128,12
    8,128,128,128,128,128,128,22
    2,1630
37  10D2 DATA 239,128,128,128,12
    8,130,135,130,128,130,135,13
    D,1605
D1  1004 DATA 148,254,190,148,21
    2,190,190,148,128,164,174,23
    5,2101
A1  1006 DATA 235,186,146,128,12
    8,227,179,152,140,230,227,12
    8,2106
2C  1008 DATA 128,146,183,237,15
    7,178,168,236,128,128,128,12
    8,1945
1E  1D10 DATA 130,135,130,128,12
    8,128,156,190,227,193,128,12
    8,1809
C5  1012 DATA 128,128,193,227,19
    0,156,128,128,136,170,190,15
    6,1930
AE  1014 DATA 136,190,170,136,12
    8,136,136,19D,190,136,136,12
    8,1832
67  1016 DATA 128,128,24D,224,19
    2,128,128,128,128,144,136,14
    D,1844
A3  1018 DATA 136,136,136,136,12
    8,128,176,176,192,128,128,12
    8,1728
54  1D20 DATA 128,160,176,152,14
    0,134,131,129,128,19D,255,2D
    1,1924
54  1022 DATA 197,255,190,128,12
    8,136,196,254,191,160,128,12
    8,2091
D9  1024 DATA 128,128,194,227,245,23
    3,239,190,128,128,162,227,20
    1,2310
7A  1026 DATA 201,237,182,128,14
    4,152,148,146,255,255,176,12
    8,2152
2C  1028 DATA 128,167,213,197,20
    5,221,184,128,128,190,255,20
    9,2225
EB  1030 DATA 201,251,176,128,12
    8,130,131,243,251,167,131,12
    8,2065
06  1032 DATA 128,182,237,201,20
    1,219,182,128,128,150,175,20
    1,2132
AA  1034 DATA 229,191,150,128,12
    8,128,16D,180,150,130,128,12
    8,1838
54  1036 DATA 128,128,192,180,18
    2,130,128,128,128,136,156,18
    2,1798
0F  1D30 DATA 227,193,193,128,12
    8,140,146,140,148,140,140,12
    8,1005
C4  1040 DATA 128,193,193,227,18
    2,156,136,128,128,136,134,16
    3,1904
D1  1042 DATA 235,143,134,128,12
    8,190,193,221,221,209,174,12
    8,2104
45  1044 DATA 128,140,186,170,17
    0,190,180,144,128,130,191,19
    1,1964
53  1046 DATA 164,162,150,156,12
    8,160,156,190,162,166,162,14
    4,1908
0F  1048 DATA 128,160,156,190,16
    2,165,191,151,128,156,190,17
    8,1985
85  1050 DATA 170,230,180,128,12
    8,136,196,190,191,149,133,12
    9,1968
82  1052 DATA 128,172,222,210,20
    2,254,188,132,128,130,191,15
    9,2116

AA  1054 DATA 132,196,252,184,12
    8,144,168,189,187,160,144,13
    6,2D20
67  1056 DATA 128,160,192,200,25
    3,185,144,128,120,130,191,19
    1,2030
2E  1058 DATA 168,156,180,144,12
    8,192,162,190,191,144,128,12
    8,1911
68  1060 DATA 128,190,158,132,14
    0,166,190,156,128,190,158,13
    2,1868
0D  1062 DATA 194,190,180,144,12
    8,220,190,178,171,190,158,12
    9,2080
D2  1D64 DATA 128,132,254,254,14
    6,162,158,140,128,156,158,16
    2,1978
22  1066 DATA 146,254,254,160,12
    8,132,190,190,140,130,134,13
    2,1998
92  1068 DATA 126,196,174,186,17
    4,186,145,128,128,140,132,15
    8,1875
F8  1070 DATA 191,164,150,128,12
    0,156,190,160,144,190,190,14
    4,1935
```

```
78  1072 DATA 128 132 142 150 17
    6 180 150 142 128 156 158 17
    6 1834
E0  1074 DATA 156 176 158 156 14
    4 162 162 156 156 182 162 13
    2 1922
44  1076 DATA 128 132 174 222 20
    8 244 190,150 128 129 202 20
    2 2117
C3  1078 DATA 160 186 144 128 12
    8 128 255 255 193 163 128 12
    8 2056
17  1080 DATA 128 208 232 254 22
    1 201 202 160 128 128 163 19
    3 2248
75  1082 DATA 255 255 128 128 12
    8 132 134,255 255 134 132 12
    8 2064
69  1084 DATA 128 140 150 191 14
    0 140 140 140 152 152 152 15
    2 1785
1C  1086 DATA 152 152 152,152 12
    8 162 188 150 145 255 190 16
    0 2024
FC  1088 DATA 128 226 191 173 16
    5 159 138 128 128 156 190 17
    7 1956
36  1090 DATA 255 161 146 128,12
    8 162 161 191,163 163 161 15
    8 2037
08  1062 DATA 128 156 190,229 25
    5 226 151 128 128 192 209 19
    0 2165
7A  1094 DATA 191 137,137,130 12
    8 188 250 195 211 250,178 12
    6 2123
E6  1096 DATA 128 164 190 191 13
    3 164 160 156 128 162 160,16
    0 2046
36  1068 DATA 191,146 128 128 12
    8 160 208 193 255 160,130 12
    6 1985
E4  1100 DATA 128 194 255 216 18
    2 226 224 128 128 192 252,16
    0 2315
1C  1102 DATA 227,224 192 128 25
    2,160 131 160 156 195 160 15
    6 2231
D1  1104 DATA 164 161-191,130 19
    5,161 190,128 128 162 190 16
    1 2111
85  1106 DATA 201 197 160 156,12
    8 166 161 191 145 143 142 12
    0 2011
88  1108 DATA 128 158-191 169 16
    5 255 222 162 162 188,190,16
    6,2216
4F  1110 DATA 157 183 226 128 19
    2 182 160 205 205,190 153,12
    8 2139
FC  1112 DATA 128 178 136 207 16
    1 191 147 129 128 188 254 19
    3 2073
50  1114 DATA 166 254 191 128 12
    8 132 158 161 224 226 190 15
    6 2177
8A  1116 DATA 128 158 161 166 19
    0 225 252 161 128 192 227 21
    5 2263
03  1118 DATA 141 220 243 227 12
    8 129 161 254 208 217 206 16
    6 2330
BF  1120 DATA 128 160,210 202 23
    7 161 187 144 140 140 140-25
    5 2134
34  1122 DATA 255 140 140 140 14
    9,146 170,170 128 128 128 12
    8,1825
A0  1124 DATA 128 128 128 255 25
    5 128 128 128 149 149 170 17
    0 1916
38  1126 DATA 149 146 170 170 23
    0 204 153 176 230 204 153 17
    9 -1
```

PROGRAM CONDENSED SET

```
62  100 REM * CONDENSED *
8A  110 SA=6220#
42  120 PRINT ESC CODE FOR SET -
    CHR$(27)CHR$( SA/256 )
23  130 B=0 FORX=0TO11 READA 0-#
    +A POKESA+X A NEXT SA-#A+X R
    EADA 1FA-#THEN130
04  140 1FA>0THEN#0INT"DATA ERRO
    # IN LINE PEEK(63)+PEEK(64)*
    256
40  100D DATA 128 128 128 128 D.
    0,0,0 128 128 223 128 1119
37  1002 DATA 0,D,0,D 135 128 13
    5 128 0 0,0 0.526
2F  1004 DATA 255,140,255 128 0.
    0,0 0,174 235 186 128 1509
D0  1006 DATA 0 0,0 0.243 136,23
    1,128 0 0 0,0 738
DC  1008 DATA 255 217,166 208,0.
    0 0 0 128 139 135 128 1376
41  1010 DATA D,0.0 D 156,162,19
    3 128 0 0 0,0 639
7E  1012 DATA 193 162,156,128,0.
    0 0 0 170 156 170 128 1263
77  1014 DATA 0 0 0 0 136 190 13
    6 128 0 0,0 0.590
82  1016 DATA 192 240 176,128 0
    0 0 0 136,136 136 128 1272
54  1018 DATA 0 0 0.0 0 128 176 17
    6 128 0 0 0 0.608
99  1020 DATA 176 140 131 128 0.
    0 0 0 191 161 161,128 1246
25  1022 DATA 0 0.0 0.164 191,16
    0 128 0 0 0,0 643
26  1024 DATA 177 166 166 128 0
    0.0 0 161 165 155,128 1249
AD  1026 DATA 0 0 0.0,143,136 19
    0 128 0 0-0 0 597
FD  1028 DATA 167 165 153,128 0
    0.0.0 156 166 153 128 1216
79  1030 DATA 0 0 0.0 126 177 14
    1 131 0 0 0 0 570
BB  1032 DATA 191 165 191,120 0.
    0 0 0-166-153 142,128 1264
E3  1034 DATA 0 0 0 0 0 128,148-14
    8 128 0 0 0 0 552
D7  1036 DATA 128 212,180 128 0.
    D 0 0-136-148 162 128 1222
63  1038 DATA 0 0 0 0-0 140,148,14
    8,128 0 0 0 0 572
73  1040 DATA 162,148 136 128 0.
    0 0.0 195 217 135 128 1249
3D  1042 DATA 0 0 0 0 0 190 217 21
    7-222 0 0,0.0 846
18  1044 DATA 186,170 190-120 0
    0 0 0 191 162 190 128-1345
4A  1046 DATA 0 0 0 0 0 190 162 16
    2 128 0 0 0 0 642
D9  1048 DATA 190 162 191 128 0.
    0 0 0 190 170 174 128 1333
41  1050 DATA 0-0 0 0 200 190 13
    7,128,0-0 0.0.655
D1  1052 DATA 222,210 254,128 0
    0-0.0 161-130 190,128 1453
FB  1054 DATA 0 0 0 0 165,186 16
    0 128 0 0 0 0 642
F3  1056 DATA 160 167 253 128 0
    0 0 0 191 136 182 128 1375
78  1058 DATA 0 0 0 0.161,161 16
    0,128 0 0,0 0 640
9C  1060 DATA 160 140 160 128 0
    0 0 0.160-130 188 128 1284
06  1062 DATA 0 0 0 0.160 162 16
    0 128 0 0 0 0 670
2E  1064 DATA 254 162 190 128,0
    0 0 0 160 162 254 128 1468
C9  1066 DATA 0 0.0.0 190 130 13
    4 128 0 0.0 0 582
A0  1068 DATA 174 170,186 128 0
    0 0 0 130 191 162 128 1266
12  1070 DATA 0 0 0 0.158,160 19
    0 128 0 0 0 0 636
C9  1072 DATA 158 160,158 128,0
    0 0 0 160 152 160 128 1264
20  1074 DATA 0 0 0 0 182,136 18
    2 128 0 0 0 0 628
2F  1076 DATA 222 208 254 128 0
    0 0 0 178 170 166 128 1454
21  1078 DATA 0 0 0 0 255,193 12
    8 128 0 0 0 0 704
70  1080 DATA 164,160 165 128 0
    0 0 0 128 193 255 128 1351
3B  1082 DATA 0 0 0 0,0 132 191 13
    2 128 0 0 0 0 583
16  1084 DATA 136 156 170,136 0
    D,0 0 128 128 128 128 1110
D0  1086 DATA 0 0 0 0.161 133 19
    1,128 0 0 0 0 643
42  1088 DATA 191 165 187 128,0
    0-0 0 191 161 161 128 1312
10  1060 DATA 0 0 0 0 191-161 15
    8 128 0 0 0 0 638
39  1092 DATA 191 165 165 128 0
    0 0 0 161 133 133 128 1234
4C  1094 DATA 0 0 0 0 191 161 18
    5 128 0 0 0 0 665
7A  1096 DATA 161 132-191 128 0.
    0 0 0 161 191 161 128 1203
E6  1098 DATA 0 0 0 0 176 161,16
    1 128 0 0 0 0 656
52  1100 DATA 191 132 187 128 0
    0 0 0 191 160 160 128 1277
92  1102 DATA 0 0 0 0 191 134 16
    1 128 0 0 0 0 644
26  1104 DATA 191 129,160 128 0
    0 0 0 158 161 158 128 1243
31  1106 DATA 0 0.0 0,191 137 14
    3 128 0 0 0 0 569
82  1108 DATA 158 161 254 192 0
    0 0 0 161 137 143 128 1404
A7  1110 DATA 0 0 0 0 167,165 16
    9 128 0 0 0 0 649
00  1112 DATA 126 191 129 128 0,
    0,D 0 156 160 161 128 1215
75  1114 OATA 0 0 0 0 156 160 15
    9,128 0 0 0 0 606
68  1116 DATA 161 152,191 128 0
    0 0 0 187 132 187 128 1246
3C  1118 DATA 0 0 0 0 135 188 13
    5 128 0 0 0 0 586
61  1120 DATA 185 165 163 128 0
    0 0 0 0 0 0 641
E5  1122 DATA 0 0 0 0.0 0 0 0 0
    0,0 0 0
8B  1124 OATA 0 0,0 0 0 0 0.0 0
    0,0 0 0
C5  1126 DATA 0 0.0 0 0 0 0 0 0
    0 0,0 -1
```

00	ø 16	@ 32	p 48	— 64	P 80
! 01	1 17	a 33	q 49	A 65	Q 81
" 02	2 18	b 34	r 50	B 65	R 82
# 03	3 19	c 35	s 51	C 67	S 83
$ 04	4 20	d 35	t 52	D 68	T 84
% 05	5 21	e 37	u 53	E 69	U 85
& 06	6 22	f 38	v 54	F 70	V 86
' 07	7 23	g 39	w 55	G 71	W 87

(08	8 24	h 40	x 56	H 72	X 88	
) 09	9 25	i 41	y 57	I 73	Y 89	
* 10	26	j 42	z 58	J 74	Z 90	
+ 11	, 27	k 43	[59	K 74	+ 91	
, 12	< 28	l 44	£ 60	L 75	92	
- 13	= 29	m 45	] 61	M 76	93	
14	> 30	n 46	↑ 62	N 78	94	
/ 15	? 31	o 47	← 63	O 79	95	

The above shows the order in which the information defining the characters is stored. This order has been chosen because it makes the easiest (therefore quickest) conversion for the driving software.

NOTE The characters from 96 to 127 cannot be re-defined. The driving software prints the normal MPS 803 characters. The characters no 96 to 127 are those printed with CHR$(224) to CHR$(255)

C64 PLUS DISK	C64	NO LOAD DIRECTORY	1/1

When you have produced a program it is wise to prevent other people from reading your disks to see what is on them. The routine presented here will allow you to stop the disk directory from being loaded into the computers memory.

The routine forces the disk drive to keep on reading the directory, without actually loading it.

P.A.Eves

```
10 PRINT "INSERT DISK - &"; A$;
20 OPEN15,8,15,"I0"
30 OPEN1,8,15,"#0":PRINT#15,"B-P"1
40 T=18:S=1:GOSUB3:T=T:S=S
50 FOR I=0TO1:PRINT#1,"B-P"CHR$(I
*24):GET#1,B$:B%=ASC(B$+CHR$(0)):
NEXT
60 IFB%(0)=18THENS=B%(1):GOTO40
70 PRINT#1,"B-P"5"0":PRINT#5,CHR
$(T);CHR$(1);
80 PRINT#1,"U2"5"0"T;S:INPUT#1,
A,A$:PRINTA$:CLOSE1:CLOSE5
```

YC WRITER

Sell your typewriter and get into serious wordprocessing with with 'YC Writer', an 80 column wordprocessor.

By Burghard-Henry Lehmann

People who know nothing of the joys and tribulations of programming a computer and who are not interested in arcade games often ask rather cynically

What good are computers anyhow?'

Of course, they do not question the use of the kind of computer the gas board for example employs. The usefulness of this becomes clear every quarter when they get their gas bill. But what good is a home computer?

Well I believe there is one good solid reason why nearly everyone should invest in a home computer: wordprocessing. Everybody who has to do any writing at all be it for work or for pleasure can benefit tremendously from using a wordprocessor. Even if all you want to do is write some letters you will not know how easy and carefree it can be until you have done it on a wordprocessor.

A wordprocessor is more than just a souped-up typewriter or even an electronic typewriter. It should really be called a 'text processor' because a good wordprocessing program goes far beyond letting you enter mere 'words'. It allows you to build up a piece of text and then restructure it in any way you like. And all this without wasting a single sheet of paper.

No more second third and fourth drafts! You start by writing from the top of your head and correct the text as you go along. A wordprocessor allows you to develop a letter, an article or even a novel from its inception to its final form all in one go without wasting time on rewriting manuscript pages which have come to look like battlefields. The savings in time and material are tremendous!

Getting Started

There is no better way to find out about wordprocessing than doing it. This is what I have written *YC Writer* for, to give you a very real taste of it.

The first thing you will notice when the program has started is that the letters are much smaller than the ordinary C 64 letters. This is because *YC Writer* uses a sort of microprint which is printed on the high-resolution screen and gives you 80 characters per screen row.

This is of course the number of characters you get on any of the Commodore printers. So the main advantage of *YC Writer* is that you'll get on paper exactly what you see on screen.

This kind of microprint may take a while to get used to depending on the kind of TV set you've got. If you are unhappy with the way things are and find it an excessive strain on your eyes do a bit of experimenting with different colours and also with different brightness and contrast settings on your TV.

To experiment with different

background and foreground colours hold down the CTRL–key and press "M". Now you will be prompted to enter the border, paper and ink colours you prefer. Type the number of the colour you want (e.g. 6 for blue – see your Commodore manual) and the colour will be changed immediately. Finally, if you are satisfied with your settings, press "Y" to return to the text, if not press "N" and the process will be repeated.

Entering Commands

Most commands to the program are given with the CTRL key held down and a single letter being entered, e.g. CTRL+L for "LOAD textfile", CTRL+S for "SAVE textfile" and so on.

Help

If you press function key 1 you will be presented with the first of the two help pages which the program incorporates. The RETURN key gets you the second help page and lets you toggle between the two pages. Function key 1 returns you to the text.

Remember all of the letters given with functions are to be entered with CTRL held down!

Information On Screen

The first three lines at the top are reserved for information. First you get the number of the line and the number of the column the cursor is on at any moment. Enter a few words and you'll see what I mean.

Next to it you get the number of words you have written so far. This wordcount is updated as you write. Later on when you start editing text it soon gets out of date. So, to get the exact wordcount press CTRL+U. This will update the number of words contained in the whole of the textfile.

Next to the number of words in the top line you see a "W", a "J" and a space in between four stars. These letters tell you which text entry mode is switched on. "W" stands for "word wrap" and "J" stands for "right hand justification". More about this and the space next to it in a minute.

The line below gives you the name of the document you are writing. When you enter the program this has the default name "no name". You can change this to a 16–letter name of your own choice by pressing CTRL+N. Now the cursor will move into the right position, ready for you to enter the document name.

There is a practical reason for this. This name will also be the filename used later on when you want to SAVE your document onto disk or tape.

The Tabulator

The third line at the top of the screen also has a practical reason beyond mere cosmetics. It shows you the tab positions on each line.

To start with there is a tab point every 5 characters. Press function key 5 and the cursor will jump forward to the next tab position. Press function key 6 and the cursor will jump backwards to the former tab position.

You can install your own tab-position anywhere on the line by pressing CTRL+T. A "/" appears on the tab-line at the top of the screen where the new tab is. Press CTRL+T again and the "/" vanishes.

If you want a completely different set-up of tab positions than the one given press CTRL+F. Now all the tab-points are erased and with CTRL+T you can make up your own tab spacings. Press CTRL+F again and the default tab positions are restored.

Entry Modes

There are really two sides to wordprocessing: you want to enter text and you want to edit it in the most convenient manner possible.

For this *YC Writer* has three entry modes: word wrap, right hand justification and insert.

Word wrap means that you can write your text as if you had one long continuous line. That is, you can ignore the end of a line and the computer does the rest. If you start a word at the end of the line it will automatically be moved onto the new line while you are writing.

For this to work there is an extra keystroke at the beginning of each line. If you enter a letter at the beginning of a new line the computer will know that this is part of a word started on the previous line and will move the whole word onto the new line. If you enter a space, the cursor won't move on, because the computer knows that the characters on the end of the line above form a complete word and are not to be moved (or "word wrapped") onto the new line. All this works of course only if you have the word wrap mode switched on. When the program starts you will find it is on, but you can switch it on or off by pressing CTRL+W.

Right Justification

The next important entry feature is right hand justification. This always works in combination with word wrap and means that the line is spaced out in such a way that it is flush with the right hand side. Like word wrap it works automatically as you write.

Again you can turn right hand justification on or off by pressing CTRL+J. But note: You can have word wrap without justification to get the "typewriter look" but you can't have justification without word wrap.

Insert Mode

The third entry mode is most useful when you want to edit the text you have written and add additional words or whole sentences.

For this move the cursor onto the paragraph into which you want to insert something. Then press CTRL+I.

Now the paragraph is reformatted by the computer, that is it is "un–justified" so that there is at least one space at the end of each line. This is necessary for insert to work properly. Don't worry about this restructuring of the paragraph! After you have done your insert and switched insert off again (as you should every time !) the whole paragraph will automatically be word wrapped and justified again!

Once in the insert mode you can enter text, but it will not overwrite other text. Instead the text to the right will be pushed along by the cursor. If there isn't enough space at the end of the paragraph it will automatically insert an empty line. So you can insert as much as you like.

Do remember to switch insert off after you have finished! Otherwise it will go on wherever you put the cursor.

If you want an empty line anywhere you can insert one by pressing CTRL+A.

This will move the rest of the textfile down by one line

Erasing

Conversely you can erase the line the cursor is on by pressing CTRL+B This moves the rest of the textfile into the line you want to be erased

If you want simply to erase one or two characters use the delete key as normal

If you want to erase a whole block of text quickly and efficiently there is a powerful block erase facility For this you first have to tell the computer the first line of the block you want to be erased and then the last line

This is called marking out a block and I mention it especially because the same procedure will also be used for marking out a block which you want to be moved or copied It works like this

Block-set

Move the cursor onto the first line of the block you want to mark out Press CTRL+G You will notice that in the information header at the top of the screen a remark has appeared for example "Blk-start 4" This is to remind you that you have marked out the beginning of a block starting at line 4

Next move the cursor to the last line of the block you want to mark out and press again CTRL+G Now the message "Blk-end 10" will appear at the top of the screen

You have now set a block starting at line 4 and ending at line 10 inclusive This is now the current block Thus a block always goes from the beginning of one line to the end of another

If for any reason you are not happy with the parameters you have given press CTRL+G again and the info at the top of the screen will be erased so that you can start again

To erase the block you have marked out simply press CTRL+K

If you want to get rid of the whole textfile and start afresh press CTRL+E Since this is a pretty final command there is a safety-catch built into it You will be asked if you are sure about erasing everything If you are not press "N" and no harm will be done If you are certain press "Y" and not only will the whole of the textfile be erased but the program will reset as if you just have started it off

Moving and Copying

If you want to move the current block to somewhere else in the textfile move the cursor to the line above the one you want the block moved to and press CTRL+O

Similarly if you want to copy the block bring the cursor to the line you want and press CTRL+H

You will notice that after block erasure and block move the messages at the top of the screen will vanish Not so after block-copy! This is so that you can copy the block you have chosen as many times as you wish But this only works of course, as long as you don't do any more editing If you do the position of the block may have changed so that you will get something different copied out!

Formating Text

At any given time you can reformat a paragraph to have it right hand justified or not CTRL+C un-justifies the paragraph the cursor is on while CTRL+D justifies it

For all this,and the insert mode to work properly the computer has to know where a paragraph starts and ends So there is an important rule A paragraph has to be started with an indent of at least two spaces!

Other wordprocessors use formatting characters to mark out the beginning or the end of a paragraph With *YC Writer* I wanted to have no distracting formating characters on screen For this to work, you have to obey the above rule A small price to pay, don't you agree?

Margins

Impressive as 80-columns on screen and on paper are, with most Commodore printers it looks rather cramped because it fills nearly the whole width of an A4 sheet This doesn't look very good if you are writing a letter you want to create a good impression

For this reason I was determined to include a margin setting facility in *YC Writer* It only works on fresh textfile before you have entered any text You have to stick with the margins you have chosen throughout the textfile and can't change them afterwards

Let's say you want a left hand margin of 10 characters Put the cursor into the right position (2 tab-positions with F5) and press CTRL+X

The margin is demonstrated graphically by the space on the left being filled and by the cursor position becoming column 0

If you now want to set a margin of 10 characters to the right again place the cursor at the appropriate position and press CTRL+Y A similar thing will happen

For technical reasons there is a rule (more rules!) to all this The left margin has to be set on an even numbered column while the right margin has to be set on an odd numbered column!

Saving and Loading

Once you have written your document the first thing you want to do it is SAVE it on disk or tape

It is a good idea to do this at regular intervals Powercuts, however brief, are not an unknown thing and it takes only a few cycles of no electricity and hours of your work may be down the drain!

Saving a document is very straight forward After you have given it the name you want as I have already described Press CTRL+S

In order to make the program work for tape as well as disk, so you don't always have to answer the annoying question "Tape or Disk?" you can switch the program into the tape or disk mode by POKEing location 8010 from Basic and then saving that version

As a matter of fact, you only have to do this POKE if you are using tape Simply POKE 8010,1 If for any reason you want to revert to disk, POKE 8010,8

The disk version of the program includes a replace facility You can use this if you have already saved a certain document and want merely to replace it with a changed version

Since there is a lot of discussion going on in the C 64 community about the safety of the replace facility and I seem to belong to the 2 percent of disk-drive owners where it isn't safe to use I have circumnavigated this sordid area by doing the replace in *YC Writer* with a combination of scratch and SAVE Better safe than sorry!

Printout

Finally, you will naturally want your document printed out

This is very easy with *YC Writer*. Simply get your printer switched on and ready and then press CTRL+P. The whole of your document will be sent to the printer as it appears on screen.

Many wordprocessors have additional print options like page-numbers, footers headers and so on. I didn't have enough time in developing this program to give you anything but a straight forward printout. I also thought that this is a nice little programming job you could easily do yourself in Basic.

Incidentally, you can quit the program at any given moment by pressing function key 7. And you can re-enter it by entering SYS 8050.

The is some 8k of memory empty for your own programs. The textfile starts in memory at location 16020.

There textfile is limited to 300 lines. Not quite enough for 'War and Peace' but remember Tolstoy's textfile was limited to one sheet of foolscap and just think of all his corrections!

Getting it all in

Entering the programs

1) Type in each of the programs presented here seperately using the SYNTAX CHECKER found on the LISTINGS page.

2) If using cassette SAVE WRITER BOOT on a seperate cassette to the other programs.

3) LOAD and RUN YC WRITER A & B. When B has finished it will SAVE a new progrom out. If using tape this should be SAVEd after WRITER BOOT.

4) LOAD and RUN YC WRITER C to YC WRITER O. When finished a second program will be SAVEd. If using tape this should be SAVEd after the program from 3.

5) To RUN the progrom simply LOAD and RUN WRITER BOOT. This will LOAD the other two parts and start the program.

```
PROGRAM  YC WRITER A

2F  10 GL=34     LN=30     SA=4915
    2
58  20 FOR L=0 TO GL CX=0 FOR D=
    0 TO 15 READ A CX=CX+A POKE
    SA+L*16+D,A NEXT D
A5  30 READ A IF A>CX THENPRINT
    "ERROR IN LINE";LN+(L*10) ST
    OP
48  40 NEXT L END
77  50 DATA 173,237,3,201 2 240
    3 76,120,31,30,173,233 3 237
    235 2023
4C  60 DATA 3,133 101,173,234 3
    237,236,3 133,102,230,101,20
    0 2,230,2369
10  70 DATA 102,165,167,133 68,1
    33 169 133,170 56,165 168,22
    9 172,133,87,2357
AC  80 DATA 133,160 133 170,176,
    6 160 00 168 169,190 179,24,
    165,170,105,2300
60  90 DATA 30,31,133,170 144,2,
    230 179 30,173,64,31,229,170
    133 73,1892
35  100 DATA 173 65,31,229 179 5
    73 144,14,24,165 170,109 57
    ,31 133,1610
43  110 DATA 170 144,229,230,179
    176,225,30,165 170,229,87,1
    33 92,165,179 2045
^8  120 DATA 229 88 133,93,24,16
    5,160,101,101 193,65,165,169
    ,101,102 133 2154
31  130 DATA 30 32 0,35 32 37 19
    3 144,10 24 173,235,3,101,10
    1,133,1431
57  140 DATA 87 173,236 3,101 10
    2,133,60,76 165 192 173 235,
    3,133 87 2067
AD  150 DATA 173,236 3 133,60 16
    5,160 133 65 165 109,133 60
    165 101 133,2224
F5  160 DATA 92,165,102 133,93 3
    2 0 35,30,173,64,31,229,160,
    133 73,1659
28  170 DATA 173,63,31 229 169 5
    ,73,176 10,24,165 168,101 16
    1,141 64 1783
78  180 DATA 31,189,189,101 160,
    141 65,31,70,236 192,24,173,
    64 31 101,1760
00  190 DATA 101,141,64,31,173,6
    5,31,101,102,141,65,31,32,37
    193 144,1612
E1  200 DATA 34 24,173,235,3 101
    101 141 235,3,173,236 3 101
    ,160,141,1960
DE  210 DATA 236 3 24,173,233 3
    101,101 141 233,3,173,234,3,
    101,102 2024
50  220 DATA 141,234,3,173,232 3
    ,240 3 76,79 37 165 171,133
    91,32,1013
15  230 DATA 100,45 76 120 31,56
    ,173,235 3 229,169,133 73,17
    3,236 3 1934
40  240 DATA 229,165,5,73 30,32,
    145 170,200,132 16,200 249,3
    2,90 193 2107
52  250 DATA 165,13,32,210 295 3
    2,210,255 96 69 70 84,69,82
    32,70 1730
ED  260 DATA 73,76,69,78 65 77,6
    9 50 32,0,169,0 76 98 193,76
    ,1209
BE  270 DATA 164 193,32 60 229,2
    4 162 12,160 0,32,240 255,16
    9 147,160 2047
53  280 DATA 193,32,30 171,169 9
    7,133,170 169 31 133,171,160
    ,0,169 32,1676
00  290 DATA 145 170,200,192 16,
    208 249,32 164 193,169,13 32
    ,210,255 32,2280
53  300 DATA 210 255 96 69 70 84
    69,82 32,70,73,76 69,78 65,
    77,1483
A7  310 DATA 69,50 32 0,169,0 13
    3,204 78 207,32 220,255,240
    245 201 2143
04  320 DATA 17 240 241 201,145,
    240 237 72 56 165 211 233,16
    ,168 104 166 2512
EE  330 DATA 211 224,16,240 50,1
    44 48 192 16 176,25,201 13 2
    40 61 32 1809
C0  340 DATA 210 255 201,20 240
    47,201 32 144 202 201,129 14
    4,2,73,128 2229
C5  350 DATA 145 170 200 192 32
    220 255 240 251,201 13,240 3
    1 201 20 240 2675
0E  360 DATA 222 201,157 240,210
    208 237 201 13 240,17,201 2
    0 240 165 201 2701
80  370 DATA 157 240,161 200 202
    130,169,32 145 170 200,152
    162 1,134 204 2489
8E  380 DATA 202 134 207,56 0 0
    0,0 0 0 0 0 0 0 0,0 639
04  390 DATA 0,0 0 0,0,0 0 0,0,0
    ,0 0 0,0 0 0 0
```

```
PROGRAM  YC WRITER B

C0  10 GL=152    LN=50     SA=4970
    9
58  20 FOR L=0 TO GL CX=0 FOR D=
    0 TO 15 READ A CX=CX+A POKE
    SA+L*16+D,A NEXT D
A5  30 READ A IF A>CX THENPRINT
    "ERROR IN LINE" LN+(L*10) ST
    OP
AC  40 NEXT L PRINT"OOOWNEDGET R
    EADY TO SAVE AND PRESS A KEY
    " POKE198,0 WAIT198,1
A2  45 POKE43,0 POKE44,192 POKE4
    5,27 POKE46,204 X=PEEK(186)
    SAVE"WRITER1  X
69  50 DATA 120,32,62 156,32 230
    ,195,32 18 156,165,0 141,152
    ,2 169 1776
51  60 DATA 165 141,20,3 169 194
    ,141,21 3,169,235,141,30 3 1
    69,197,1609
61  70 DATA 141 39,3 32,152,195,
    88,96,173 192,2,240,3,76,41,
    195 1668
60  80 DATA 220,104,2 240 79,169
    ,30,205,184,2,240,3,76 41,19
    5,169 2045
1F  90 DATA 1 141,168,2,165 251
    141 169 2 165,252,141,169 2,
    173 167 2166
```

```
8A 100 DATA 2 141,130,2,133 251
   ,173,100 2,141,131 2,133 252
   32 214 2027
96 110 DATA 195 160,7 177 251,1
   53 133 2,174,171 2,208 5,8,2
   40 76 2023
60 120 DATA 241 194,9,15 145,25
   1,136,16 234 32,222 185 169
   20 141 104 2204
CC 130 DATA 2 76 31 185 169 0 1
   41,100 2,173,130 2,133,251 1
   73,131 1515
A6 140 DATA 2,133 252,160 7,185
   ,193 2,145 251 136,16 248,16
   9 70 141 2110
25 150 DATA 184 2,173 100,2,133
   251 173,101 2 133,252,32,23
   4,255 165 2360
E9 160 DATA 204 208,41 130,205
   208 37 169 20 133,208 164,21
   1,70 207 174,2454
01 170 DATA 135 2 177 209 170 1
   7,230 207 133,206 32,36 234,
   177 243,141,2355
0F 180 DATA 105 2,174 134,2 165
   ,206 73,128 32,28 234 165 1
   41 16,1536
9C 190 DATA 240,10 160 0 132 19
   2 165 1 9 32,200 0 165 152,2
   00 6 1720
00 200 DATA 165 1,41 31,133 1 3
   2 135 234 202,189 119,2 201
   148,208 1634
24 210 DATA 23 198 130 173 17 2
   00 41 32,208 6 32,152 195 70
   145 195 1503
CC 220 DATA 32,182 195,189,14,3
   2 202 241 76 126,234,173,2,2
   21 9 3 1511
C5 230 DATA 141 2,221 173 17,20
   0 9 32 141 17,208 173,0,221,
   41 252 1856
6A 240 DATA 141 0 221 169 00 14
   1,24 208,96 173 2,221 9 3 14
   1 2,1687
33 250 DATA 221,173 17 200 41,2
   23,141,17,200,173,0 221 41,2
   52 9,3 1940
1E 260 DATA 141 0 221,169,21,14
   1 24 200 96 120,165,1,41,253
   133,1 1735
94 270 DATA 96 165,1 9,2,133 1,
   00 96 169 0 133,251,169,224,
   133 1670
52 280 DATA 252 160 0 169 0 145
   ,251 165 251,201,63 208 7 16
   5,252 201 2450
39 290 DATA 255 208,1 96,230 25
   1 208 2,230 252 76,240 195 1
   69 0 133 2546
99 300 DATA 251,169 204,133,252
   ,160 0 173,33 208 41,15 133,
   253,173 134 2332
6B 310 DATA 2 10 10 10 10,41,24
   0 5 253 145,251 166 251,224,
   232,208 2050
F1 320 DATA 7 100 252 224 207,2
   00 1 96,230 251,208,2 230,25
   2 76,38 2448
F2 330 DATA 195 169 0 141,167,2
   169 224 141,168 2,169,0,141
   ,169 2 1850
D4 340 DATA 169 204,141 170,2,1
   69 0 141 171 2 96,169,0,141
   173 2 1752
08 350 DATA 32 214,195,100 123,
   133 251 169,202 133,252 24,1
   4,172,2 46 2131
06 360 DATA 173,2 24,14 172,2,4
   6 173,2,24,173 172,2 101,251
   ,133,1464
77 370 DATA 251,173,173 2,101 2
   52,133 252,173,167 2 133,253
   173,168,2 2488
AE 380 DATA 133 254,173,171 2 2
   00,60 100 0 162,4,177,253,41
   ,15 141,1562
21 390 DATA 172,2,177,251,41,24
   0 13,172,2,145,253,230 253,2
   00,2 230 2381
FC 400 DATA 254 177 253,41 15,1
   41 172,2,177 251,10,10 10 10
   ,13,172,1700
16 410 DATA 2 145,253,230 253 2
   00 2 230 254 230,251 200,2,2
   30 252 202 2852
6A 420 DATA 200 201,230 171,2 3
   2 222,105 76,35,157 100,0,16
   2,4 177 2000
D5 430 DATA 253 41,240,141,172
   2 177 251 74,74,74,74,13 172
   ,2,145,1805
22 440 DATA 253 230,253 200,2 2
   30,254 177 253 41,240 141,17
   2 2 177 251 2084
03 450 DATA 41 15 13,172,2,145,
   253 230,253 200 2 230 254 23
   0 251 200,2007
FE 460 DATA 2 230 252 202 208 2
   01,165,253 141 167 2 165 254
   ,141,160 2 2553
1G 470 DATA 200 171 2 32 222 15
   5,173 169,2,133,251,173,170
   2 133 252,2206
AE 480 DATA 160 0,173,134,2 10
   10,10 10 141 172,2,177,251,4
   1,15 1300
CE 490 DATA 13,172,2,145,251,17
   3,171 2 200,0,236,185 2,200,
   3,230 2003
GF 500 DATA 178,2,30 169,0,133,
   251,24 105,64,133,253,169,22
   4,133 252,2170
C6 510 DATA 100 1 133 254 32,21
   4 185 100,0,177 253,145,251,
   165 253 201,2535
45 520 DATA 63 200 6,165 254,20
   1 255,240,15 230 251 200 2,2
   30 252 230 2010
36 530 DATA 253 200 2 230 254,7
   6 102 157 32 222,185 169 0,2
   30 251 200,2629
B4 540 DATA 2 230 252 145,251,1
   00 251,224,63,208,6,166,252,
   224 255,240,2935
E0 550 DATA 3,70 138 197,169 0,
   133,251 24,105 40 133 253 16
   9 204 133 2028
C5 560 DATA 252,185,0,133,254,1
   77 253,145,251 165,253,201,2
   32 200 6 165 2000
22 570 DATA 254,201 207,240,15,
   230 251 208 2 230,252,230,25
   3 208 2,230 3013
3E 580 DATA 254,76,170,197,169,
   0 141 167,2,169,254,141,168
   2,169 192,2279
CF 590 DATA 141 169,2,169,207,1
   41,170 2,169 0 141 171,2 96,
   141,172,1093
CF 600 DATA 2,72,138,72,152,72
   165 154 201,3,240,8,104,168
   104,170 1025
4E 610 DATA 104,76,202,241,169,
   1 141,152,2,173,186,2,240,30
   169,0 1920
45 620 DATA 141,186,2 173,190 2
   133 251,173,191,2 133 252,1
   60 7 185 2181
2C 630 DATA 193,2,145 251 136 1
   6 248 169 51 141,184,2 173 1
   72,2 201,2046
69 640 DATA 255 200 8 169 126 1
   41,172 2 76 79 189,201,224 1
   44 9 56 2050
11 650 DATA 233 64,141,172,2 76
   79 189 201,152,144 6,56 233
   96 141 2034
20 660 DATA 172 2 201 13,208 6
   32 246 201 76,238 201,201 17
   ,208 117 2139
D0 670 DATA 173 189 2 201,254 1
   44,73 173,167,2 201 0 144 66
   ,173 171 2112
03 680 DATA 2 141 170 2 173 167
   ,2,141 174 2 173 168,2 141,1
   75 2,1643
00 690 DATA 173,169 2,141 176 2
   ,173 170 2 141,177,2 32 00 1
   67 173 1010
4A 700 DATA 170 2,141,171,2 173
   174 2 141,167,2 173,175,2 1
   41 168 1012
77 710 DATA 2 173,176 2 141 169
   2 173 177,2,141,178,2,76,20
   7 198 1011
A6 720 DATA 24,173,167 2 105 64
   141 167,2 173,168 2 105 1 1
   41 169 1603
09 730 DATA 2 24,173,169 2 105
   40 141,169,2,173,170 2 100 0
   ,141,1418
0C 740 DATA 170,2 76,230 201 20
   1 15 208 6,32 62,198 76,230,
   201,201 2127
#5 750 DATA 28,248,3,76 240 189
   173 167 2 141,176 2,173,160
   2,141,1931
93 760 DATA 177 2 173 169,2 141
   ,178 2,173,170,2,141,179 2,3
   2 246 1700
72 770 DATA 201 173,178,2 141,1
   67 2 173,177,2 141 180 2 173
   178,2,1070
AE 780 DATA 141,189 2 173,179,2
   141,170 2,102 6 160,6 32,21
   4 195 1700
F3 790 DATA 173,171,2,200 77 17
   3 167,2,201,0,200,13 173,160
   ,2 201,1935
4D 800 DATA 224 200,6 32,222,19
   5,76 238 201,50,173 189 2 29
   3,1,141,2177
66 810 DATA 169 2 173,178 2 233
   ,0 141 176 2 56 173 176,2 23
   3,0 1710
9A 820 DATA 141 176 2 141 167 2
   173 177 2 233,0 141 177 2 1
   41 168 1043
E4 830 DATA 2 173,176 2,133 253
   173 177 2 133,254 177,253 4
   1 248 141 2330
F8 840 DATA 102 2,173 174 2 133
   251 173 175 2 133,252 140 1
   72 2 56 2022
C9 850 DATA 165,251,233 6 133 2
   51 165 252,233,0 133 252 177
   251,24 46 2574
A1 860 DATA 172 2,42 46 172 2 4
   2 46 172,2 42,46 172 2,42 46
   1048
C0 870 DATA 172 2 145 251 46,17
   2 2 46 172 2 46 172 2 46,172
   2 1450
```

```
6C  800 DATA 165,251,205,176,2,2
    00,208,165,252,205,177,2,208
    ,193,173,171,2753
D7  810 DATA 2,208,9,177,253,41,
    15,13,182,2,145,253,230,174,
    2,208,1922
9A  820 DATA 3,230,175,2,238,178
    ,2,208,3,238,177,2,202,240,1
    1,173,2088
8F  830 DATA 171,2,240,3,76,111
    ,165,76,34,165,32,222,165,206
    ,171,2,1858
7C  840 DATA 240,6,238,171,2,238
    ,171,2,76,230,201,201,29,208
    ,59,173,2253
87  850 DATA 171,2,208,6,238,171
    ,2,76,230,201,206,171,2,24,1
    73,167,2056
A2  860 DATA 2,105,0,141,167,2,1
    73,168,2,105,0,141,168,2,238
    ,165,1551
F5  870 DATA 2,208,3,238,178,2,1
    73,167,2,201,64,208,10,173,1
    68,2,1791
4D  880 DATA 201,255,208,3,32,80
    ,197,76,230,201,201,141,208,
    0,32,240,2320
A0  890 DATA 201,76,230,201,201,
    145,208,53,102,225,230,166,2
    ,144,9,208,2477
5E  900 DATA 41,174,167,2,224,64
    ,144,34,56,173,167,2,233,64,
    141,167,1053
4C  910 DATA 2,173,168,2,233,1,1
    41,168,2,56,173,169,2,233,48
    ,141,1704
A8  1000 DATA 169,2,173,170,2,23
    3,0,141,170,2,76,230,201,201
    ,147,202,2133
30  1010 DATA 12,32,230,165,32,1
    0,196,32,82,196,76,230,201,2
    01,148,240,2101
80  1020 DATA 3,76,114,201,173,1
    67,2,141,170,2,173,168,2,141
    ,177,2,1710
58  1030 DATA 173,169,2,141,178,
    2,173,170,2,141,179,2,32,240
    ,201,173,1504
EF  1040 DATA 176,2,141,187,2,17
    3,177,2,141,188,2,173,178,2,
    141,165,1014
23  1050 DATA 2,173,179,2,141,17
    0,2,173,167,2,133,201,173,16
    8,2,133,1071
3A  1060 DATA 252,168,0,140,178,
    2,162,8,32,214,195,173,171,2
    ,240,9,1932
7D  1070 DATA 177,251,141,182,2
    41,15,145,251,177,251,24,110
    ,178,2,168,2047
D8  1080 DATA 110,172,2,106,110,
    172,2,106,110,172,2,106,110,
    172,2,145,1599
5F  1090 DATA 251,110,172,2,110,
    172,2,110,172,2,24,165,251,1027
7D  1100 DATA 105,0,133,251,165,
    252,105,0,133,252,165,251,20
    5,174,2,200,2405
F4  1110 DATA 200,165,252,205,17
    5,2,208,193,173,171,2,240,17
    ,173,176,2,2354
24  1120 DATA 133,251,173,177,2,
    133,252,173,182,2,41,240,149
    ,251,238,176,2565
C7  1130 DATA 2,208,3,238,177,2
    173,176,2,133,251,173,177,2
2C  1140 DATA 230,174,2,208,3,23
    0,175,2,140,172,2,173,171,2,
    240,9,1949
85  1150 DATA 177,251,141,182,2,
    41,15,145,251,208,240,3,76,2
    30,200,32,2102
DD  1160 DATA 222,105,70,230,201
    ,201,157,208,60,173,171,2,24
    0,6,200,171,2535
4E  1170 DATA 2,76,230,201,173,1
    87,2,201,0,208,10,173,168,2,
    201,224,2046
08  1180 DATA 208,3,76,230,201,2
    30,171,2,56,173,167,2,233,0,
    141,167,2004
BA  1190 DATA 2,173,168,2,233,0
    141,168,2,56,173,169,2,233,1
    ,141,1844
30  1200 DATA 169,2,173,170,2,23
    3,0,141,170,2,76,230,201,201
    ,160,208,2146
C6  1210 DATA 5,169,32,141,172,2
    201,161,144,5,169,120,141,1
    72,2,201,1045
4A  1220 DATA 32,144,30,201,120,
    176,26,56,233,32,141,172,2,3
    2,82,196,1690
2A  1230 DATA 173,167,2,201,64,2
    00,10,173,168,2,201,255,220,
    3,32,80,1947
42  1240 DATA 197,169,0,141,192,
    2,76,245,197,169,0,141,174,2
    ,169,224,2102
E2  1250 DATA 141,175,2,162,6,17
    3,175,2,205,160,2,144,12,220
    ,31,173,1773
26  1260 DATA 174,2,205,167,2,24
    0,2,176,21,24,173,174,2,105,
    64,141,1672
54  1270 DATA 174,2,173,175,2,10
    9,1,141,175,2,232,70,2,202,1
    73,172,1807
A0  1280 DATA 2,201,140,240,72,2
    01,20,240,60,173,174,2,141,1
    67,2,173,2024
77  1290 DATA 179,2,141,160,2,10
    9,0,141,171,2,169,0,141,105
    ,2,169,1621
D5  1300 DATA 204,141,170,2,232
    ,202,240,20,24,173,169,2,105
    ,40,141,169,2034
19  1310 DATA 2,173,170,2,105,0
    141,170,2,70,82,202,173,167,
    2,201,1602
E6  1320 DATA 64,144,10,173,168,
    2,201,255,208,3,32,82,197,90
    ,0,0,1633
26  1330 DATA 0,0,102,102,6,96,0
    5,0,0,05,245,95,05,39,170,
    110,0
E0  1340 DATA 115,220,234,242,79
    ,87,4,164,169,112,30,0,0,0,1
    0,60,1554
20  1350 DATA 60,33,60,17,17,30,
    0,150,240,144,2,39,34,0,0,0
    852
28  1360 DATA 3,52,0,7,0,0,0,0,6
    ,90,17,34,60,136,105,159,713

70  1370 DATA 153,96,30,34,34,11
    2,105,10,72,240,113,39,10,64
    ,15,121,1276
98  1380 DATA 241,16,116,71,17,3
    6,10,70,153,96,241,34,60,64,
    105,150,1564
DE  1390 DATA 153,96,105,151,10,
    64,6,96,6,96,6,96,6,100,10,7
    2,1885
45  1400 DATA 66,16,0,112,112,0
    ,132,33,36,128,37,10,32,32,10
    5,107,1046
F0  1410 DATA 136,112,0,113,117
    ,112,60,117,85,112,0,116,60,1
    12,17,117,1402
01  1420 DATA 85,112,0,117,116,1
    12,3,60,100,64,0,117,87,23,6
    0,117,1105
E0  1430 DATA 85,42,32,34,34,32
    ,32,34,34,36,69,86,101,00,68,
    60,985
E1  1440 DATA 68,96,0,150,153,14
    4,0,0,117,85,60,0,117,85,112,0
    ,117,1333
C7  1450 DATA 87,60,0,117,87,17
    0,116,60,64,0,116,113,112,00
    ,116,1149
A9  1460 DATA 60,112,0,85,85,112
    ,0,85,85,32,0,153,159,144,0,
    85,1205
90  1470 DATA 37,00,0,05,07,23,0
    ,113,36,112,110,102,102,112
    37,79,1123
40  1480 DATA 60,240,115,51,51,1
    12,2,114,34,34,0,75,64,0,0,1
    5,979
6C  1490 DATA 240,0,37,07,85,00
    ,101,87,85,95,117,60,69,112,1
    01,05,1450
D5  1500 DATA 85,96,116,71,60,11
    2,110,71,60,64,117,71,85,112
    ,85,87,1424
0A  1510 DATA 05,00,114,34,34,11
    2,114,34,42,224,05,05,101,00
    ,60,60,1361
B6  1520 DATA 60,112,159,153,153
    ,144,157,223,107,144,117,85,
    85,112,117,87,2103
0F  1530 DATA 60,64,117,85,85,35
    ,117,87,101,00,110,71,17,112
    ,114,34,1303
40  1540 DATA 34,32,05,85,05,112
    ,05,05,05,32,153,153,159,144
    ,85,82,1468
5A  1550 DATA 37,00,05,85,34,32
    113,10,60,112,102,111,246,10
    2,165,165,1555
7D  1560 DATA 165,165,102,102,10
    2,102,0,250,170,100,255,119
    51,17,255,255,2270
47  1570 DATA 255,255,0,0,0,0,0,
    0,0,0,0,0,0,0,0,510
```

```
PROGRAM: YC WRITER C

14  10 GL=49      LN=50      SA=5731

50  20 FOR L=0 TO GL CX=0 FOR D=
    0 TO 15 READ A CX=CX+A POKE
    SA+L*16+D,A NEXT D
A5  30 READ A IF A><CX THENPRINT
    "ERROR IN LINE",LN+(L*10) ST
    OP
40  40 NEXT L END
3F  60 DATA 76,105,22,76,206,23
    32,206,23,160,0,140,232,3,13
    2,161,1617
5E  60 DATA 132,160,200,140,236,
    3,160,0,177,160,201,32,208,0
```

```
    ,200 204 2253
20  70 DATA 58,31,200,244,240,93
    ,132 183 132,184,177,120,201
    ,32 208 38 2329
29  80 DATA 204,58,31 176 48 133
    ,73 200 177 160 201 32 208,2
    2,132 183 2846
4F  90 DATA 164 184 145 120,164
    183,177 160 201 32 200 10,20
    0 204 57 31 2304
78  100 DATA 220 244,240 17 165
    73 164 184 145,120 230 183,2
    30 184 164 183 2702
F2  110 DATA 204,57 31 208 197,1
    73 232 3 220 183 164 184 204
    57 31 176 2232
85  120 DATA 51,169 32,145,168 2
    00 204 57,31 200,240,173 230
    ,3 240 3 2170
0D  130 DATA 32,120,45,24 165,16
    8,105 57 31 133,100,144 2,23
    0,169 230 1053
A7  140 DATA 31 160 0 177,120 20
    1 32 200 7 220 177 168,201,3
    2 240 3,2065
20  150 DATA 76,121,22,58 173,20
    31,240,210 177 160,201 32,2
    40 3 136 1994
3D  160 DATA 200,242,162 0,200,1
    77 160 157,65 3,169,32,145,1
    00 232 200 2329
F8  170 DATA 204 57 31,200,240 1
    57,65 3,232 134,101 230 232
    3,76 222,2283
CC  180 DATA 22 164,184 204 58,3
    1,176 10,169 32,145,160 200
    204 57 31 1655
40  190 DATA 200 240 165,184,133
    ,103 24,101 101,200,57,31,14
    4 66 56,173 2159
60  200 DATA 58 31,225 101,120 1
    77 160 201 32,240 3 136 200
    247 138 103 2394
C0  210 DATA 200,162 0 177 160 1
    57 234 159,201 32 200 0 200,
    177 160 201,2452
27  220 DATA 32 240 12,130 169,3
    2 145,160 232 220,204,57,31,
    200 220 169 2263
02  230 DATA 32,157 234,159 232,
    134 102 24 165,183,101,101,1
    33,184 200,2 2211
CF  240 DATA 133 184,164 183 177
    160 184 184,145,185 190,184
    158 183,16 242,2691
1C  250 DATA 186 101 164 184 202
    189 65 3,145 160 202 136 16
    ,247,166 102 2416
15  260 DATA 240 15 169 234 159,
    157 65,3 202 16 247,165,182,
    133 101 169 2361
95  270 DATA 0 133 102,240,3 220
    ,232 3 76 222,22 165,166 133
    166 165 2116
0E  280 DATA 167 133,169,58,165,
    160 229 172 133 160,176,2,19
    0,165 165 171 2441
01  290 DATA 133 91 165,160 205
    69 31 200 7 165,165,205 70,3
    1,240 32 1989
D0  300 DATA 160 0 177,120,201,3
    2 200 7 200,177 120 201,32,2
    40 17,56 2044
C0  310 DATA 165,160 237,57,31 1
    33 169 176,2 198,165 120 91
    76 225 23,2121
F8  320 DATA 96,127,127,127,127,
    127,127,127,127 127 127,127,
    127 127 127,127 2001
60  330 DATA 127,127 127 127,127
    ,127 127 127,127 127,127,127
    ,127,127 127 2032
3A  340 DATA 127 127,127 127,127
    ,32,102 22,169 0,133,101,133
    ,102,141,232 1962
89  350 DATA 3,165,168 133,73 16
    5,165,133,74,24,165 73,169 5
    7,31,133,1675
D1  360 DATA 73 144,2 230,74,120
    ,0,177,73 201,32 200,10,220
    177 73 1034
A4  370 DATA 201,32 200,3 230 23
    2,3,172,57,31,177 120,201,32
    ,240 44,2039
6D  380 DATA 136,177,120 201 32
    240 37,173 64 31,240,32 162,
    0,177 160 2842
84  390 DATA 201,32,240 4 232,13
    6,200 246,134,101,162,0,200,
    177,160,157,2476
60  400 DATA 65,3,169 32,145 160
    ,200 232,200,101,200,241,173
    ,232 3,220 2420
89  410 DATA 3,32,20,30 32,120,4
    5,24,165,160,109 57,31 133,1
    68 144,1347
31  420 DATA 2,230 169,230,91,16
    0,0,177,120,201,32,220,63,20
    0,177,120 2276
C6  430 DATA 201,32,200 10,165 1
    01 240,3,76,146,25 76 97,25,
    160 0 1645
0F  440 DATA 177,168 201 32 220,
    30,173,20,31,200,3 76,20 24,
    32,104,1791
D7  450 DATA 200,132,183 174,57
    31 164 183,177,120,164 184,1
    45 160 230 183 2543
09  460 DATA 230,184,202,208,241
    ,172,50,31,169 32,145,160 17
    3,20,31,240,2362
5D  470 DATA 20,165,101,240 20 1
    72 50,31,186,101,177,160,201
    32 200,79 2235
07  480 DATA 136 202 208,246 177
    160 201 32,200 3 130 200 24
    7 132 183 24 2511
19  490 DATA 165,183,101,101,133
    184 164,183,177,126,164,184
    145 120 120,184,2662
07  500 DATA 158 183 16,242,120,
    101,164,184 202,189,85,3 145
    160 202,130,2444
F4  510 DATA 16,247,169,0,133,10
    1,120,182,240,17,189,234,159
    ,157,65 3,2150
03  520 DATA 202,16 247,165,182
    133,101 169,0 133,162,76,20,
    24 96 58 1530
EC  530 DATA 173 58,31,225,181,1
    60,177 120 201,32,240,3,130,
    200 247 200 2462
3F  540 DATA 162 0 177,120,157 2
    54,0 0,0 0,0,0 0,0,0,0 894
```

```
PROGRAM  YC WRITER D

14  10 @L=45     LN=50     SA=6521

5B  20 FOR L=0 TO @L CX=0 FOR D=
    0 TO 15 READ A CX=CX+A POKE
    SA+L*16+D,A NEXT D
A5  30 READ A IF A><CX THENPRINT
    "ERROR IN LINE",LN+(L*10) ST
    OP
48  40 NEXT L END
7E  50 DATA 159,169,32,145,168,2
    32,208,204,57,31 208,240,134
    ,102,56,173,2330
9C  60 DATA 58,31 229,101,133,10
    3 76,34,25 165,168,72,105,10
    9,72 165,1926
ED  70 DATA 91,72,32,224,46 104,
    133,91,104 133,169,104 133,1
    60 58,185,1025
DD  80 DATA 168,237,57,31,133,16
    0 176,2,120,169,32,20,30,189
    ,51 32,1742
DA  90 DATA 120,45,24,165,168,10
    9,57,31 133,168,144,2,230 10
    9,230,91 1952
4C  100 DATA 162 0,160 0 189,65,
    3,145,120 200,232 220 101 20
    0,245,32 2218
41  110 DATA 106 45,96,165,166,1
    33,160,165,167 133 109,58 16
    5,168,220,172 2303
20  120 DATA 133 160 176,2,198 1
    69,172 58,31,177,160,201,32,
    220 32 132,2057
10  130 DATA 184,136,132,183,164
    ,183,177,188 164,184,145,160
    ,198,184 188,183 2751
FD  140 DATA 164 184,196 172,208
    ,238,169,32 145,168 169,140
    32 210,255 76,2555
91  150 DATA 120,31,127 127,127,
    127,127 127,127,127 127,127,
    127,127,127,127 1928
C4  160 DATA 127 127,127 127,127
    ,127,127,127 127 127 127,127
    ,127,127,127 2032
CA  170 DATA 127,127,127,127,127
    ,127,127,127,127,127 127,127
    ,127,127,127,127 2032
1C  180 DATA 127 127 127 127,127
    ,127,127,127,127,127,127,201
    ,1 200 6 32 1845
76  190 DATA 227 46 76,120 31 201
    1,2 200 3,76,186,49 201,3 20
    0 11 1520
D9  200 DATA 173 80,31 200,3 32,
    93,22,76,120,31 201 4 200 11
    ,173 1468
4F  210 DATA 60 31 200,244 32,56
    ,24 76 120 31,201 5,200,3,76
    158 1575
2C  220 DATA 47,201 6,208 3,76 6
    2 58,201,7 200 3 76 50,35 20
    1 1494
AC  230 DATA 0 200 3,76,83 35 20
    1 9 200,3 76 246,27,201 10 2
    00 1612
A3  240 DATA 3 76 121,27,201,11,
    208 3,76,79,37 201,12,200,3,
    76 1342
20  250 DATA 71,52,201,13,200 3
    76 4 61,201,14 200,3,76 89 2
    0 1300
B9  260 DATA 201,15 200,3,76,99,
    35 201 16,200 3,76,104,36,20
    1 10 1580
A8  270 DATA 220,4,169 1,200 6,2
    01 19 200,7,169 0 133,255 76
    ,74,1738
CE  280 DATA 52,201 20 208 3 76
    97,47 201 21,208 6,32 94 51,
    76 1393
```

```
03  290 DATA 120,31,201 23,240,2
    4,201,24,208,3,76,102,20,201
    ,25,208 1735
67  300 DATA 3,76,165,20 201,26,
    208,3,76,0,35 76,120 31 173,
    65 1307
GH  310 DATA 31,240,47 100,0 141
    66,31,141,07,31 32,60,40,16
    9,0 1265
33  320 DATA 141,171,2,169,184,1
    41,167 2,169,224,141,168 2,1
    69 23 141 2014
6F  330 DATA 169,2,169,204,141,1
    70,2,169,32,32,72 40,32,63,4
    0,76,1413
0A  340 DATA 126,27,169,1,141,66
    ,31,133,177,32,60 40,100,0 1
    41,171 1404
4C  350 DATA 2,169,184,141,167,2
    ,169,224,141,168 2 165,23,14
    1,169,2 1473
59  360 DATA 169,204,141,170 2,1
    69,215,32,72 40,32,63 40 76,
    120 31 1576
D0  370 DATA 173,67,31,240,44,16
    9,0,141,07,31,32,00,40,169,0
    ,141,1405
96  380 DATA 171,2,100,200,141,1
    07,2,169,224,141,100,2,169,2
    5 141 189,2060
AD  390 DATA 2 169,204,141,170,2
    169 32,32,72,40,32,00,40,76
    120 1304
A0  400 DATA 31,169,1,141,07,31,
    141,60,31,133 177,100,0,141,
    171,2,1471
4B  410 DATA 169 184,141,167 2,1
    69,224,141 169 2,169,23,141,
    168,2,169 2040
9B  420 DATA 204 141,170,2,100,2
    15,32,72,40 169 200 141,167,
    2,169,224,2117
35  430 DATA 141,168,2,169,90,14
    1,169,2,169,204,141,170,2,16
    9,0,141,1013
32  440 DATA 171 2 169 202,90,72
    ,40 32 63 40,76 120 31,173,0
    0,31 1322
EE  450 DATA 240,47 32 56 24,169
    0,141,60,31 32,60 40 169,0,
    141 1250
4F  460 DATA 171,2,169,210,141,1
    67,2 169,224,141 100,2,100,2
    7,141 168,2070
7D  470 DATA 2 169,204,141,170 2
    169,32,32,72 40,32 63,40 76
    ,120 1364
9E  480 DATA 31,169,1,141,60 31,
    32,60 40,169,0 141,171,2 169
    ,216,1441
D0  490 DATA 141,167,2,169,224,1
    41,168 2 169 27 141,169,2,16
    9,204 141 2036
26  500 DATA 170 2,169,901,32,72
    40,32,63,40 32,99 22,76,120
    31,1201
00  510 DATA 32,60,40,169,184,14
    1,167,2 169,225,141,168,2,16
    9,45 141 1775
97  520 DATA 169,2,169,204,141 1
    70 2 169,0,141,171,2,169 00,
    133,179 1500
9F  530 DATA 169,31 133,179 169,
    16 133,211 100,165 141,20 3,
    169 194,141 2043
96  540 DATA 21 3 00 32,95 153,1
    20 32,63 40,76,120 31,0,0,0,
    914

PROGRAM: YC WRITER E

01  10 BL=50    LN=00    SA=7350

50  20 FOR L=0 TO 8: CX=0 FOR D=
    0 TO 15 READ A:CX=CX+A POKE
    SA+L*16+D,A NEXT D
A5  30 READ A IF A><CX THENPRINT
    "ERROR IN LINE";LN+(L*10):ST
    OP
48  40 NEXT L END
45  50 DATA 76,194,20 76,35,29,7
    6,00,29,76,177,29 32,67 29,2
    40 1203
52  60 DATA 3,76,120,31,165 172,
    240,245 74,176,246,24 173 03
    ,31,200 2051
0F  70 DATA 240,101 172 133,172,
    00,173 57,31 229,172 141 57,
    31,100 136 2069
FE  80 DATA 140,50,31 165 172,14
    1,63,31 74,133,73 170 24 173
    ,59,31,1530
54  90 DATA 105,0,141,59 31 144,
    3,230,60,31,230,61 31 200 3
    230,1592
93  100 DATA 62,31,202 200 231 5
    6,165,166 229,172,133,166 17
    6,2 198,167 2364
7A  110 DATA 165,0,133 172,32,23
    9,44 32,00,90,76 120 31 32,6
    7 29,1285
0S  120 DATA 240 3,76 120 31,165
    ,172 201 79 240,247 74,144 2
    44 164,172 2372
SE  130 DATA 140 50,31 200 140,5
    7,31 32,177 29,76 126 31 56
    173 64,1415
94  140 DATA 31,237 69,31,133,73
    ,173 56,31 237,76,31,5,73,24
    ,1 1500
BB  150 DATA 00,165,171,00,173,1
    07 2 133 160 133,07,173 100,
    2 133 169 2036
C0  160 DATA 133,00 100 73,56 16
    5,160 233 0 133,100,133,07,1
    76 4 100 1909
00  170 DATA 169,100,00,202,200
    230 169 22,133,74,166 73,169
    170,165 0 2235
E0  180 DATA 145 07,200 192,8 20
    0 248 24,165 07,105,8,133 07
    144 2,100 1
90  190 DATA 230 00,202,200 231
    24 165,160 105,64 133 160 13
    3,07 165 169 2340
9E  200 DATA 105 1,133,169,133 0
    0 150 74,200 200,00,173 167
    2,133,160 2056
20  210 DATA 133 07,173,160 2,13
    3 169,133 00 24,105,160 105,
    0,133,160 1657
3C  220 DATA 133 07 144,4,230,16
    9 230,00 169 22,133,73,56 16
    9,00 225,2016
01  230 DATA 172,237,63 31 74 17
    0,165,05 160,0,145,07 200,10
    2,0 200 2001
60  240 DATA 245,24 165,07,105,0
    ,133,07 144,2,230 00 202 200
    231 24 1007
25  250 DATA 165,160 105 64 133
    160 133,07 165 169,105,1,133
    100,133,00,1569
20  260 DATA 150 73 200 200 96 1
    27,127,127 127,127,127 127 1
    27,127,172 00,2140

76  270 DATA 31,177 160,201 32 2
    40 1 00 162,0 134,105 134,10
    7,232 134,2114
45  280 DATA 186 160,0,177 160 2
    01 32,200 13 177,160 201 32
    200 7 200,2130
C3  290 DATA 204 57 31 200,244 9
    6 177 160 201 32 200,2 230 1
    65 200 204 2447
0E  300 DATA 57,31,200 242 165,1
    05 240 207,172 50 31,132,104
    ,136,177 160 2393
40  310 DATA 201 32 200,5 230 10
    6 136 200 245,132 103,56 165
    ,105,225 100 2587
0E  320 DATA 133,105 197,106,176
    22 162,1 56,165 106 229,105
    133 106,197 2393
A2  330 DATA 105 144,3 232 200,2
    42 134,107 166 105 200,2,166
    ,106 164 103 2595
00  340 DATA 177 160 164 104 145
    ,160,201 32 240,7,198,104 15
    0,103 76 132 2457
02  350 DATA 30 198 104 164 104
    169 32 145 160 100 104 150,1
    03 165 107 240 2629
00  360 DATA 29 134 73 166 107 1
    65 106 240 5 169 106 70,163
    30 202 240 2300
F2  370 DATA 11,164 104 169,32 1
    45,160 158 104 202 200 245,1
    66 73 202 200 2559
CC  380 DATA 165 56 255 255 255
    255 255 255 255 255 255 255
    255 255 255 3055
A0  390 DATA 255,255 255 255,255
    255 255 255 255 255 255,255
    255 255 255 255 4900
3E  400 DATA 255,255,255,255 255
    255 255 255 255 255,255 255
    255,255 255 4890
41  410 DATA 255 255,255 255,255
    ,255 255 255 255,255 127,127
    ,127 127 127 3312
C9  420 DATA 127 127,127,127 127
    127 127 127 127,127,127 127
    127,127 127 2032
4F  430 DATA 127 127 127,127,127
    127,127 127,127 127 127,127
    ,127,127 127 2032
90  440 DATA 127 127 127 127 127
    ,127 127 127 127 127,127,12,
    0 0 0 41 1450
80  450 DATA 0,122,0 00 78 152,2
    27 120 204 6 160 0 1 1 0,160
    ,1390
F2  460 DATA 62 155 159 127,0,12
    7 127 64 03 50 70 78 32,78 6
    5 77 1375
58  470 DATA 69 32,32 32 32 32 3
    2 32,32,32 127,127,127 127,1
    27 127 1119
10  486 DATA 127 127,127,127 127
    ,127 127,127 127 127,127,127
    ,76 120,31 76 1836
2E  490 DATA 226 31,76,14,32 76,
    61 32 76 70 32 169,254 141,3
    3 200 1531
78  500 DATA 169 6 141 134 2 141
    32 200 169 120,141 130 2 16
    0,7,169 1747
E4  510 DATA 0 153,193 2 136,16
    250 32,125,194,32 125 38,173
    69 31,1569
DA  520 DATA 133 166 133,73,141,
    64,31 173 70 31,133,167 133,
    74 141,65,1720
```

```
5C  530 DATA 31,32,78 40 32 81 4
    0 166 80,140 57 31 130,140 5
    6 31 1167
06  540 DATA 169,192,141 59 31 1
    69 227 141 60,31,169 120 141
     61 31 169 1911
3B  550 DATA 204 0,0,0,0,0,0,0 0
     0 0,0 0 0 0 204
```

PROGRAM YC WRITER F

```
A7  10 0L=56     LN=50     SA=0151
50  20 FOR L=0 TO 6L CX=0 FOR D=
    0 TO 15 READ A CX=CX+A POKE
    SA+L*16+D,A NEXT D
A5  30 GEAD A IF A><CX THENPRINT
    "ERGOG IN LINE" LN+(L*10) ST
    OP
40  40 NEXT L END
ED  50 DATA 141 62 31,169,0,141
    63 31,76,14 32,173,55,31 133
     166 1310
3C  60 DATA 173 56 31 133,167 17
    3 54,31 133 171 173,53,31,13
    3 172,173 1057
F0  70 DATA 51 31 133,175,173,52
     31 133 176,173 49 31,133,17
    3 173 50 1737
3A  80 DATA 31,133,174,120 32,13
    2 194 32 228 255,240 251,201
     133 200,3 2367
45  90 DATA 76 246 54 201 135,00
    8,3 76 06 44,201 136 240 24
    261 139 2070
DC  100 DATA 200 3 76 05 44 172
    141 2,192 4 200 3,76 04 26 2
    01,1529
6D  110 DATA 134 240 212 76,130,
    32 32 66 229 32 152,195 32 1
    62 195,169 2110
D1  120 DATA 48,141 20 3 169,234
    141 21 3,169 202 141,30 3,1
    69 241 1744
66  130 DATA 141,33,3 165 130 14
    1 55 31 165 167 141,56 31,16
    5,171,141 1770
76  140 DATA 54 31,165 172 141 5
    3,31 165 175,141 51 31 165,1
    76 141 52 1744
5A  150 DATA 31,185 173 141,49,3
    1 165 174 141 50 31,30,255,2
    55 255 255,2267
A0  160 DATA 255 253 255 76,144
    32 76 61,33,201 13 208,3,76,
    84 34 1014
61  170 DATA 201 17 208 3,76,211
     33 201 18,240 45 201,20,208
    3 76 1782
9C  180 DATA 119 46 201,32 208 3
    76 208 50 201 145 208 3 76
    37 34 1639
55  190 DATA 201 148 208 3 76 22
    0 25 201,157 206 3 76 30 33
    201 134 1932
69  200 DATA 208 3 76 123 31,76
    254 41 120 173 55,31 141,167
     2 173 1670
64  210 DATA 60 31 141 130 2 173
    61 31 141 169 2 173 62 31 1
    41 170 1556
CF  220 DATA 2 169 0 141 171,2 0
    6 56 165 166 229 172,133 166
     176 2 1030
C2  230 DATA 130,167 130,171 240
    ,15 56,165 166,237,57,31,133
    ,130,176,2,2146
49  240 DATA 138 167,202,208 241
    ,56 165,175 229,171,133 175,
    176 2,158 176,2672
FD  250 DATA 169,0 133,171 133,1
    72,32,236 44,32 238,44 76,12
    0 31,164,1766
0D  260 DATA 175 200 11 164,176
    200,7,164,172 208,3 76,120 3
    1,164,166 2053
E8  270 DATA 200 2 138 167,138 1
    66 138 172 164 172 192,255 2
    40 3 76 202,2613
77  280 DATA 33 165,171,208 37,3
    2,30 40,32,216 39,169,0 133
    51 165,1597
4F  290 DATA 130,133,138,165,167
    ,133 165,56,165,180,237,30,3
    1,133 130,176,2293
9C  300 DATA 2 130,169,32,186 45
    ,120,76,147,33,198 171 120 5
    6,173,167 1083
BB  310 DATA 2,233,64,141 167 2,
    173 160 2,233,1,141,160 2 56
    173 1726
76  320 DATA 169 2,233,40,141,16
    9,2,176 3,200,170 2,173,57,3
    1,74,1640
3F  330 DATA 170 24,173,167 2,10
    5 0 141 167 2 144,3 230,100,
    2,230 1752
7C  340 DATA 169 2 200 3 238 170
    ,2 208 200,231 00 169 157 32
     210 255,2344
27  350 DATA 173,30,31,133,172 1
    30 175 200,2 130,176 130 175
    ,32,236,44,2176
00  360 DATA 76 205,33 32 210,25
    5,32 239 44 76,100,31 173,72
    ,31 133 1752
AB  370 DATA 74,66,173 71 31 237
     57 31,133 73,176,2,130 74 5
    6 165 1007
00  380 DATA 73,225 130,133,75,1
    65,74 229 167 5 75,144,46 16
    0 0,177 1910
A3  390 DATA 130 133 73,24,105 1
    30 109 57,31 133,166,144 2 2
    30 167 165,1831
09  400 DATA 171 201 21 200 6 32
     159 34 76,25 34 230 171 169
     17 32 1626
0B  410 DATA 210,255,230 175 200
    2 230 176,32 236 44 76,120
    31,165 175,2365
48  420 DATA 208 7,165,176 208 3
    ,76 120 31 130,0,177,166 133
     73 56,1759
75  430 DATA 165,106 237,57,31,1
    33 166 176 2 150,167,165,171
     200 6 32 2080
5C  440 DATA 217 34,76 83,34,100
     171 169 145,32,210 255,165
    175,200 2,2179
6D  450 DATA 130 176,180,175 76
    31,34 173,72 31,133 74,56,17
    3 71 31 1782
3E  460 DATA 237 57 31 133,73 17
    6 2 198 74 56 165,73 225 130
     133 75 1070
D8  470 DATA 165 74 229 167,5 75
     144 69,56 173,57 31 229 172
    ,24,101,1771
78  480 DATA 165 133 130 144 2,2
    30 167 165,171,201 21 208,27
     165 172,72 2210
90  490 DATA 169 0 133 172 32 19
    9,34 104,170 240 10 120 169
    157 32 210,1953
6F  500 DATA 255 202 208 250 88
    76 180 34 230 171 32 142,23
    0 175 200,2302
A0  510 DATA 2 230 176 169 0,133
     172,32 238 44 32,236,44,76,
    120 31 1735
4D  520 DATA 32,60 40,32 66,40 3
    2 116 39 32,63 40 169,21 133
     91 1006
66  530 DATA 200 16 32,60,40 32,
    30 40 32 216 39 32 63 40 169
    0 1085
6A  540 DATA 133 91 165 167 133
    169 165 166 133 168 56 165 1
    88,229 172 133 2413
C7  550 DATA 169,0,0,0,0,0,0,0 6
    0 0,0 0 0 0,0 160
```

PROGRAM YC WRITER D

```
6E  10 0L=52     LN=56     SA=0952
50  20 FOR L=0 TO 0L CX=0 FOR D=
    0 TO 15 READ A CX=CX+A POKE
    SA+L*16+D,A NEXT D
A5  30 GEAD A IF A><CX THENPRINT
    "EGROR IN LINE" LN+(L*10) ST
    OP
40  40 NEXT L END
75  50 DATA 170 2,198 168 32,106
     45 96 0 72,130 72 152,72,21
    6 166 1000
57  60 DATA 93 56,165,87 229 89,
    165 88 229 90 144,24 188,6,2
    32 196 2047
6F  70 DATA 92 200 3,202 240,46
    177 87,145 89 200 208 242,23
    0 88 230 2407
30  80 DATA 90 76 23 35 164,92 1
    30 101,88 133 00 138,24,101
    90 133 1514
00  90 DATA 00 232,152 240,8 136
    ,177 87 145 89,76 50 35,198
    00 198 2009
21  100 DATA 56,202 208 241 104
    160 104,170,104 40 96,127,12
    7 127 127,127,2162
FD  110 DATA 127 127 75 102,35 7
    6,5 30 76,230,39 76,32,36 17
    3,237 1479
F8  120 DATA 3 200 52 165,1 141,
    237,3 165,167 141,236 3 56,1
    65 166,1913
92  130 DATA 229,172,141,235 3 1
    76 3 200,236,3 32 60,40 32 1
    21 36 1725
6E  140 DATA 32,147,36,169,92 16
    0,36 32 30 171 166,175 165 1
    76 32,235 1824
2D  150 DATA 109 32 63 40 76,120
     31 261 2 240 67 169,2,141 2
    37 3 1613
61  160 DATA 165 167 141 234 3 5
    6 173 58,31 229 172 24 101 1
    30,141 233 2054
1A  170 DATA 3 144 3 230 234,3 3
```

```
    2 60 40 32 121,36 24 173 167
    2 1312
D3  100 DATA 105 200 141 167 2 1
    44,3,238 168 2,173,165,2 105
    ,26 141 1794
DF  150 DATA 169 2,144 3 238 176
    2 169 104 160 36 76 143,35,
    169 0 1626
08  200 DATA 141 237,3,32 60 40,
    32 121,36,32,147,36,162 35 1
    69 32 1315
04  210 DATA 134 170 32 72 40 16
    6 170 202 208 244 76 153 35
    169 0 141 2029
80  220 DATA 232 3 56,173 235 3
    237 233 3 133 73 173 236 3 2
    37 234 2264
C6  230 DATA 3 5 73 176 7 76 0 1
    92 169,1 200,237 32 80 40 32
    1301
06  240 DATA 121 36 32 147 36 16
    2 30 169,32 32 210 255 202 2
    08 250 24,1946
14  250 DATA 173 167,2 105 200 1
    41 167,2 144,3,238,168,2 173
    169,2 1064
0E  260 DATA 105 26 141 169 2 14
    4 3 230,170 2 169,114,180 36
    32 30 1541
07  270 DATA 171 76,153 35,154 7
    6,76 45,03 04 65,02 04,80 32
    0 1313
7F  280 DATA 154 76 75 45 69 70,
    80 50,32 0 157 02 02 79 02 3
    3 1250
35  290 DATA 0,169,04 141 167,2,
    169 225 141,168 2 169 40 141
    169 2,1705
C2  300 DATA 105,204 141,170,2,1
    65 0 141 171,2 80 24 173,167
    2 105 1736
43  310 DATA 136 141,107 2,144,3
    230 160,2 173,160,2 105 15,
    141,169 1775
A9  320 DATA 2 144,3,230,170,2 9
    6 253 255,255 255 255 255,25
    5,255,255,2940
D3  330 DATA 173 64,31 205,69 31
    208 11 173,65 31,205 70,31,
    208,3,1570
F8  340 DATA 76,120,31 169,0,32
    169,255 169 4,170 160,255,32
    186,255 2103
5F  350 DATA 32 192 255,162 4 32
    201,255 169,13 32,210 255,1
    73 69,31 2005
0A  360 DATA 133 100 173 76 31 1
    33 169,160 0 165 145 201,127
    ,240 72,174 2161
E4  370 DATA 63 31 240 0 169 32,
    32 210 255 202,208 250 169 1
    7,32 210 2128
20  380 DATA 255 177 160 32 210
    255 200 204,57 31,208 240,17
    3 57 31 201 2459
FF  390 DATA 80 240 5,169 13 32
    210 255 24 165 160,105,57,31
    133 160 1059
E2  400 DATA 144,2,230 169 160 0
    56 173 64 31 229 160,133,73
    173 65 1070
4A  410 DATA 31 229 169 5 73,176
    178,169,13,32,210 255,169 4
    ,32 135 1940
67  420 DATA 255 32 204 255,76 1
    20 31 173 237 3 201,2 240 3
    76 120 2020
AF  430 DATA 31 173,233 3 133 87
    ,173,234,3,133,88,230 87 208
    ,2,230 2048
F3  440 DATA 88,173 235,3 133,89
    ,173 236,3 133,90,56,165 87,
    237,235,2136
70  450 DATA 3 133 181 165,88,23
    7 230,3 133,182 56 173,64,31
    ,225 87,2001
B8  460 DATA 133,73 173 65 31,22
    5 88 5 73 144 30 56 173 64,3
    1 229 1597
39  470 DATA 87,133 92,173 65,31
    225,88 133,93,56 165 92 225
    ,101 133 1900
0F  480 DATA 73,165,93,229,182,5
    ,73 176 8,165 101,133 92,165
    ,182 133,2055
CD  490 DATA 93 32 0,35 56,173,6
    4 31 237,233 3,133 73,173,65
    31 1432
99  500 DATA 237 234,3,5 73 176
    15 173,235 3,141 64 31,173 2
    30 3 1802
D3  510 DATA 141,65 31,76 17 38
    56 173,64 31,225 101,133 73,
    141 64 1513
E4  520 DATA 31,173,65,31 225,10
    2,133 74 141,65,31,160,0,160
    ,182 240 1503
22  530 DATA 12 169 32 145 73 200
    0 200 251,230 74,202 208,246
    166 101,240 2637
EF  540 DATA 8 169 32,145 73,200
    202,208 250 165,167 133 165
    56 165 166 2300
6A  550 DATA 229 172 133,160,170
    ,2 150 165 165 171 133,91,24
    0 16 80,165 2204
1A  560 DATA 168,237,57 31,133 1
    80,176,2,150 160,180,91 200
    240,32,180,2200
51  570 DATA 45 76 96 36 0 0 0 0
    ,0,0 0,0,0 0,0 0 252
```

PROGRAM YC WRITE0 H

```
AD  10 0L=50    LN=50    SA=5053
    
00  20 FOR L=0 TO 0L CX=0 FOR D=
    0 TO 15 READ A CX=CX+A POKE
    5A+L*16+D,A NEXT D
A6  30 READ A IF A><CX THENP61NT
    "ERROR IN LINE",LN+(L*10) ST
    OP
40  40 NEXT L END
0F  00 DATA 169,176,100,30,32,30
    ,171,169,1,100,39,32,30 171
    169,13,1500
C6  00 DATA 32,210,255,162 00 10
    9,19,39,157,0,50,202,16 247,
    169,0,1627
F8  70 DATA 100,50,32,30,171 102
    ,0,105 100,39 157 00 31 232,
    224,10 1673
0E  00 DATA 200,245,80,204 73 70
    69,00,32 45,32,32 32,105 75
    ,76 1500
0E  50 DATA 65 77,70 00 32 45 32
    32,32,215,75,82 00,03 50,32
    ,1092
39  100 DATA 40,32,32,32,32 32 3
    2,32 32,32,32 32,32,32,32 42
    530
00  110 DATA 32,215,32,42,32 202
    ,32,42,32,32 32,42 32,32 32,
    32 895
7E  120 DATA 32,32 32 32,32 32,3
    2 32,42 32 190,49,61,200,197
    ,204 1239
AA  130 DATA 200 32,42 0 196 79,
    67,85 77,69,76,84,50 32,70 7
    9,1264
16  140 DATA 32,70 65,77 69,0 22
    1,46,46,46,46 50 46,46 46 46
    ,960
62  150 DATA 221,46 46 46 46,50
    46,46,46 46,221,46,46 46,46,
    50,1110
0G  160 DATA 46,46 46,46 221,46,
    46,46,46,50,46 46,46,46 221,
    40,1030
30  170 DATA 46,46 46 50,46 46,4
    6,46,221 46,46,46,46 40,46,4
    6,935
A2  180 DATA 40,40 221 46,46,40,
    46,00 46,40 46 40 221 46 46,
    40 1030
EA  190 DATA 46,00,40 46 46 40 0
    ,70,79 32 70,65,77 69 32 32
    ,830
00  200 DATA 32 32 32 32 32,32 3
    2,32,214,155,109,0,133,07,16
    9,229 1452
73  210 DATA 133,00 185 192 133
    09 165 227,133 90,162 21 160
    0 177 07 2030
52  220 DATA 145 09,200 208,245
    230 00 230 90 177,07 145,09
    200 152 64 2403
03  230 DATA 200 247,24,105,07,1
    05 64,133 07 144,2 230,00 24
    165 09,1002
C8  240 DATA 105 64,133 09,144 2
    ,230 90,202 200 208 32,222 1
    95 90 255 2276
55  250 DATA 255 255 255,205 255
    ,255,255 255 255 255,255 255
    ,255 255,255,255 4000
40  260 DATA 255 255 255,255 255
    ,255 255,255 255 255 255,32
    214 155 169 132 3607
F5  270 DATA 133 07 169,252,133,
    00 169 0 133 09 169 254 133,
    90 162 21 2002
43  280 DATA 160 0 177 07 145 05
    200 208 249 230 00 230 90 1
    77 07 145 2362
FE  290 DATA 09 200 192 64 208 2
    47 190 00 190 90 56 165 07 2
    33,64 133 2312
2S  300 DATA 07 165,80 233,1 133
    08 56 165 05,233,64,133,05
    165 90 1079
3F  310 DATA 233 1 133 90 202 20
    0 201 32 222 155 96 127,127,
    127 127 127,2240
C4  320 DATA 127,127,127 127 127
    ,127 127,127 127 127,127,127
    127,127 127,76 1961
E4  330 DATA 88 40 76 132 40 76
    238 40 76,24 41 76 127 41 76
    ,150 1342
G2  340 DATA 41,76 179,41 76,221
    ,41 76 175 40 76,01 41 120 1
    69 49 1502
0D  350 DATA 141 20,3 169,234 14
    1 21 3,173,167,2 141 60,3 17
    3 168 1619
0A  360 DATA 2 141 61,3 173 165
```

```
    2,141,82,3,173,170,2,141,83,
    3 1309
C7  370 DATA 173,171,2,141,84,3,
    96,120,169,165,141,20 3,169,
    154,141,1772
10  380 DATA 21,3,173,60 3,141,1
    67 2 173,61 3 141 160 2,173
    62,1353
31  390 DATA 3,141,169,2,173,03,
    3,141,170,2 173 04,3,141,171
    2 1421
0D  400 DATA 00 96 120,169,160,1
    41 20 3 169 154 141 21,3,173
    ,60 3 1566
4D  410 DATA 141,107,2,133,251,1
    73 61 3 141,160,2 133,252,17
    3 62 3 1605
43  420 DATA 141 169 2 173,63 3,
    141 170 2 173,64,3,141 171,2
    32 1450
C0  430 DATA 214,195,160,7,177,2
    51 153 193 2 138,16 240,32,2
    22 195,00 2265
2D  440 DATA 96 169 1 141 192 2
    173 167 2 141 190 2,133,251
    173,160 2001
07  450 DATA 2 141 191,2,133 252
    160,7,105,193,2,145 251,136
    ,16 240 2064
C9  460 DATA 165 33 141 184 2,16
    9,0 141 192 2 60 165 251,141
    188,2 1076
2E  470 DATA 185,252 141 120,2,1
    73 167,2 141,160 2,133 251 1
    73 160,2 2151
45  480 DATA 141 191 2,133 252 3
    2 214,195,160,7 177,251 153,
    193 2 174 2277
73  490 DATA 171 2,200 5,9 240 7
    6,72 41 9 15 145 251 130,16,
    234 1630
5F  500 DATA 32 222 195,90 109 1
    ,141,192,2 173,160 2 240 30
    105,0 1050
17  510 DATA 141,186 2 173,160 2
    ,133 251 173 191 2 133 252 1
    60 7 185,2101
08  520 DATA 193,2,145,251 136,1
    6,240,169,33 141,184,2 169,0
    ,141 192,2002
5F  530 DATA 2 96,201,120,144,5,
    56 233,120,200,3,60,233,32,1
    41 172 1839
04  540 DATA 2,152 72 32,60 160,
    184,160 96 169 192,133 251,1
    69,227 32 2003
3D  550 DATA 232 0 0,0,0 0,0,0 0
    0 0 0 0 0 0 0 236
```

```
PROGRAM: YC WRITER 1
```

```
2E  10 BL=50    LN=60    SA=1065
    4
5B  20 FOR L=0 TO 0L CX=0 FOR D=
    0 TO 15 READ A CX=CX+A POKE
    SA+L*16+D,A NEXT D
A5  30 READ A IF A><CX THENPRINT
    "ERROR IN LINE",LN+(L*10) ST
    OP
48  40 NEXT L END
12  50 DATA 195,169,120,133,251,
    169,204,32 16,160,160,7,169,
    0 153,193 2167
93  60 DATA 2,160,10,250,60,60,1
    73,71,31,237,69,31,72,173,72
    ,31,1510
09  70 DATA 237 70,31 170,100,0,
    169,32,145,73,200 200 251,23
    0,79 202,2252
CF  80 DATA 208 246 104 240 9 17
    0 169,32,145,73,200,202 208,
    250,58,160,2512
39  90 DATA 0,132,171,132,91,132
    ,175,132,176,132,172,132,173
    ,138,174,140,2150
3A  100 DATA 00,31,140,237,3,200
    140,66,31,140,67,31 132,177
    96,255,1014
E1  110 DATA 76 4,42,76 232,43,1
    33,75 165 166,205,71 31 200,
    10,165,1702
2C  120 DATA 167,205,72,31,200 3
    ,76,120 31,165,75,201,25 240
    ,26,172 1021
10  130 DATA 00,31,240,3,32,82,4
    0,184,172,200,0,172,60,31,24
    0 3,1548
E0  140 DATA 70,160,42,160 0,165
    ,75,145,166,230,160,200,2,23
    0 167,60,2077
D1  150 DATA 165,160,237,84,91,1
    33,77,105,107,237,65,31,5,77
    144 10,1700
32  160 DATA 105,75,201,20,240,1
    0,105,166,141,84,31,165,167,
    141,85 31,1056
6E  170 DATA 238 172,165 172,205
    ,57,31,200,75,169,0,133,177,
    105,171 201,2331
0E  180 DATA 21,200 36 105,75 20
    1 29 240,6,32 72,40,78 120,4
    2 32 1403
00  190 DATA 66 40 32 116,39,105
    166,133 160 165 167,133,109
    ,160 21,133 1602
0E  200 DATA 91 32,160,45 76,156
    ,42,232 171,165,75,32 210,25
    5,32,232 2032
F7  210 DATA 43,109,0 133,172 32
    ,235,44,230,175 208 2,230 17
    6 32 236,2121
E6  220 DATA 44,70,120,31,165 75
    32,210 255,32,238 44,76,120
    ,31,172 1782
02  230 DATA 60,31,240 7 160,1 1
    32 177 76,45,42,164,175,200
    4,164 1650
13  240 DATA 176,240 15,201,32 2
    00 14 164 177,200 7,160,1 13
    2 177 76 1960
6D  250 DATA 120 31 76,45 42,133
    ,179 164 177 240 3,76,49,42,
    32 232,1649
82  260 DATA 44,165,167,133 169,
    56,165 168 225,172 133,160,1
    76,2,198 169 2312
F2  270 DATA 56,165,168,237 57,3
    1,133 169 176,2,168,169,160,
    177 177 2065
59  280 DATA 201,32,208,3,200 20
    0 247 177 160 201,32,240 9,2
    00 204 57,2327
15  290 DATA 31,208 244,76,45,42
    165 168 133,83 165,167,133
    60 165,85 2012
6D  300 DATA 208 2,198 60,198 85
    162 0 160 0 177 85,201 32,2
    40 11 1657
E0  310 DATA 165 85 200,2,168 90
    198 85,232 200 239,134 178
    32 232 43 2337
E2  320 DATA 165 167,133 181,56
    165 166 229,170 133 160 170
    2 150 181,162,2470
F1  330 DATA 0,166,170 240,17,17
    7 180 145 165 169 32,145 100
    169 23,32,2025
67  340 DATA 210 255 200 202,208
    239 230 178 165 179 145 166
    169 23 32 210,2017
FC  350 DATA 255,24 165,166 101
    170 133 168 144 2 230 167 24
    165 172,101 2193
0F  360 DATA 170 133,172 32,235
    44 56,165 166 237,84 31 133,
    77 165 167,2059
CC  370 DATA 237 65 31 5 77,144
    10 165 166 141 84,31 165,167
    141,65 1674
3A  380 DATA 31 105 1,133,177,10
    5,167,133,169 56,165 186 229
    ,172 133 168 2234
A7  390 DATA 176 2 100 169 168 1
    71 134 91,32,100 45,56 105 1
    60 237 57 2053
60  400 DATA 31,133,168 176 2 19
    0 169,173 67 31 240 3 32 20
    30 166 1639
A3  410 DATA 171 202 134 91,32 1
    06 45,78 120,31 120 169,0 14
    1 171,2 1691
00  420 DATA 173 59 31,141,167,2
    ,173 60 31,141,168,2,173 61,
    31,141 1554
01  430 DATA 169 2,173 02,31,141
    ,170 2,100 171,240,149,24 17
    3 87 2 1842
9F  440 DATA 195,84,141,167 2,17
    9 160,2 105 1,141,160 2 24,1
    73 169 1605
00  450 DATA 2 105 40 141 109,2,
    144,3,230 170 2 202 200 222,
    60 96 1832
F1  460 DATA 127 127 127,127 127
    ,127 127 127 127,127,127 127
    127,127,127,2032
F7  470 DATA 127,127,127,127 127
    ,127 127,127 127 127,127 127
    ,127 127,127,2032
E3  480 DATA 127,127 127,127,127
    ,127 127,127,76 82 44 70 150
    44 24 173 1655
C2  490 DATA 50 31 105 83 31,133
    73 24 165 172 105 83 31,170
    ,169,0,1421
90  500 DATA 50 201 46,240,2 208
    ,11,220,73 240,23,120 0,50 2
    01,46 1868
73  510 DATA 208,16 169 29 32 21
    0 255,230 166 260 2,232 167
    230 172 232 2556
DC  520 DATA 208 225 32,239 44 7
    6,120,31 24 165 172 105 63 3
    1 170 236 1949
3C  530 DATA 63 31 240 230,189 0
    ,50,201 46 240 2 200 12 236
    63,31,1950
C5  540 DATA 240 224 105,0 60 20
    1 46,200 217 169,157,32 210
    255,165 166 2529
7C  550 DATA 208 2 138 167 138 1
    66,168 172 202 208,226 240 1
    97,0 0 0 2302
```

PROGRAM: YC WRITER J

```
12  10 DL=51     LN=50     SA=1150
    0
58  20 FOR L=0 TO DL:CX=0:FOR D=
    0 TO 15:READ A:CX=CX+A:POKE
    SA+L*16+D,A:NEXT D
A5  30 READ A:IF A>(CX THENPRINT
    "ERROR IN LINE";LN+(L*10):ST
    OP
40  40 NEXT L:END
79  50 DATA 76,242,44,76,61,45,3
    2,90,40,169,24,141,167,2,169
    ,224:REM 1572
87  60 DATA 141,168,2,169,3,141,
    169,2,165,204,141,170,2,169,
    0,141,171:REM 1791
8A  70 DATA 171,2,160,175,232,25
    8,7,24,165,176,125,1,208,2,1
    65,176:REM 1903
01  80 DATA 32,205,125,165,170,2
    9,22,165,175,201,10,176,7,1
    69,32,32,156:REM
D8  90 DATA 210,255,200,0,201,10
    0,176,5,169,32,32,210,255,32
    ,83,40:REM 1534
FE  100 DATA 96,32,68,40,165,72,
    141,167,2,165,224,141,168,2,
    169,5:REM 1681
52  110 DATA 141,169,2,169,204,1
    41,170,2,169,0,141,171,2,165
    ,172,208:REM 2051
00  120 DATA 165,0,32,205,125,16
    6,172,251,10,170,5,169,32,32
    ,210,255:REM 2022
FA  130 DATA 32,83,40,96,127,127
    ,127,127,127,127,127,127,127
    ,127,127,127:REM 1795
70  140 DATA 127,127,127,127,253
    ,255,255,255,255,255,255,255
    ,255,255,253,255:REM 3504
3C  150 DATA 255,255,253,255,255
    ,255,255,255,255,255,255,255
    ,255,255,253:REM 4076
90  160 DATA 255,255,255,255,255
    ,255,255,253,255,255,253,255
    ,253,255,255,253:REM 4074
46  170 DATA 253,255,253,255,255
    ,255,255,255,76,109,45,76,21
    3,45,76,21:REM 2777
28  180 DATA 46,32,150,45,24,165
    ,160,125,57,31,133,150,144,2
    ,230,165:REM 1709
F1  165 DATA 230,91,105,91,251,2
    ,200,233,96,169,192,141,167
    ,2,169,227:REM 2404
D4  200 DATA 141,160,2,169,120,1
    41,169,2,169,204,141,170,2,1
    66,91,240:REM 2035
4D  210 DATA 34,24,173,167,2,150
    ,64,141,167,2,173,160,2,105,
    1,141,169:REM
63  220 DATA 160,2,24,173,169,2,
    105,40,141,169,2,144,3,230,1
    70,2:REM 1552
FC  230 DATA 202,208,222,169,0,1
    41,171,2,96,165,91,251,0,144
    ,74,201:REM 2097
3C  240 DATA 22,176,70,32,60,40
    ,32,103,45,173,63,31,240,39,1
    73,57:REM 1498
69  250 DATA 31,201,80,240,32,12
    0,173,63,31,74,170,24,173,16
    7,2,105:REM 1606
E6  260 DATA 0,141,167,2,144,3,2
    30,168,2,238,169,2,208,3,230
    ,170,96:REM
25  270 DATA 2,202,208,231,96,16
    0,0,177,168,132,73,32,72,40,
    164,73:REM 1022
D3  280 DATA 200,204,57,31,208,2
    41,32,84,40,96,127,127,127,1
    27,127,127:REM
98  290 DATA 127,127,127,127,127
    ,127,127,127,127,127,127,127
    ,175,200,11,164,211:REM
02  300 DATA 176,208,7,164,172,2
    08,3,76,120,31,56,173,57,31,
    225,172:REM 1903
37  310 DATA 133,170,125,168,133
    ,87,133,69,133,168,165,167,1
    33,90,133,88:REM 2161
91  320 DATA 133,169,165,89,208
    ,2,190,90,196,89,160,0,177,16
    6,145,89:REM 2570
43  330 DATA 230,196,178,208,247
    ,169,32,145,89,90,165,160,22
    9,172,133,168:REM 2555
12  340 DATA 176,2,196,169,165,1
    71,133,91,32,150,45,165,166,
    208,2,198:REM 2107
C0  350 DATA 167,198,168,169,157
    ,76,141,32,255,255,255,255,2
    55,255,255,255:REM 3140
90  360 DATA 255,255,255,255,70,
    238,46,76,90,47,173,85,31,13
    3,90,133:REM 2176
56  370 DATA 88,24,173,64,31,133
    ,87,165,57,31,133,25,144,2,2
    30,90:REM 1485
02  380 DATA 165,85,141,64,31,16
    5,90,141,85,31,168,0,177,87,
    145,25:REM 1640
60  390 DATA 125,87,250,2,198,90
    ,198,87,165,89,208,2,198,90,
    198,25:REM 2072
25  400 DATA 165,90,197,169,208
    ,230,165,25,197,160,250,224,1
    60,0,169,32:REM 2471
80  410 DATA 145,160,250,204,57
    ,31,250,240,32,180,45,90,169,
    167,133,169:REM 2240
A4  420 DATA 90,165,168,229,172,
    133,168,176,2,150,168,56,165
    ,180,237,64:REM 2324
47  430 DATA 31,133,251,165,169,
    237,65,31,5,251,144,1,96,166
    ,171,134:REM 2058
BC  440 DATA 91,32,230,46,90,32,
    68,40,165,125,141,107,2,169,
    225,141:REM 1770
4F  450 DATA 160,2,169,68,141,16
    9,2,169,254,141,170,2,24,165
    ,172,125:REM 1807
21  460 DATA 63,31,133,73,150,17
    6,4,160,0,240,2,160,1,140,17
    1,2:REM 1402
1C  470 DATA 170,240,27,24,173,1
    67,2,150,8,141,167,2,144,3,2
    30,168,177:REM
DA  480 DATA 2,208,169,2,208,3,2
    30,170,2,202,208,231,168,73,
    125,0:REM 2101
D0  490 DATA 90,251,46,240,5,169
    ,46,76,184,47,169,47,157,0,5
    ,32:REM 1515
01  500 DATA 210,255,32,63,40,76
    ,120,31,32,123,31,32,68,229,
    24,162:REM 1528
E7  510 DATA 12,160,4,32,240,255
    ,169,247,160,47,32,30,171,16
    9,3,133:REM 1070
04  520 DATA 211,169,24,160,48,3
    2,90,171,32,220,255,240,251,
    251,70,240:REM 2370
01  530 DATA 7,201,89,208,243,76
    ,114,31,76,117,31,62,65,65,5
    0,89:REM 1566
24  540 DATA 32,84,79,32,83,82,6
    5,83,65,32,87,72,79,76,69,32
    ,:REM 1042
DD  550 DATA 84,69,88,84,70,73,7
    6,69,33,13,13,0,10,32,65,85
    ,65:REM
43  560 DATA 69,32,89,79,85,32,0
    ,3,85,82,69,0,0,0,0,0,0:REM 705
```

PROGRAM: YC WRITER K

```
31  10 DL=49     LN=66     SA=1232
    6
58  25 FOR L=0 TO DL:CX=0:FOR D=
    0 TO 15:READ A:CX=CX+A:POKE
    SA+L*16+D,A:NEXT D
A5  30 READ A:IF A>(CX THENPRINT
    "ERROR IN LINE";LN+(L*10):ST
    OP
40  40 NEXT L:END
7C  50 DATA 83,32,40,89,47,70,41
    ,32,146,0,127,127,127,127,12
    7,127,133:REM
73  60 DATA 127,127,127,127,127,
    127,127,127,155,166,133,83,1
    33,160,165,167,226:REM
90  70 DATA 133,84,133,109,105,9
    4,133,86,56,165,83,225,172,1
    33,87,133:REM 2097
7C  80 DATA 160,176,4,136,80,136
    ,169,165,171,133,81,133,90,1
    33,76:REM 2079
AD  90 DATA 115,40,24,165,87,105
    ,2,133,87,144,2,230,90,206,5
    0,31:REM 1555
0E  100 DATA 24,165,87,109,50,31
    ,133,87,144,2,230,90,230,50,
    31,230:REM 1749
00  110 DATA 91,230,86,160,0,177
    ,87,201,32,208,9,250,177,87,
    251,32:REM 1998
23  120 DATA 208,208,240,71,160,
    2,177,87,201,32,208,150,208,
    177,87,201:REM 2477
0E  130 DATA 32,208,131,177,87,2
    01,32,208,14,208,204,87,31,2
    50,244,230:REM 2334
4B  140 DATA 87,200,40,230,90,20
    0,36,165,91,72,165,169,72,16
    5,169,72:REM 2066
C0  150 DATA 165,66,133,169,24,1
    25,87,150,2,133,160,144,2,23
    0,169,32:REM 1966
0E  160 DATA 224,46,104,133,165,
    104,133,160,104,133,91,230,9
    7,200,2,230,217:REM
10  170 DATA 90,94,165,87,225,15
    5,133,89,165,90,229,167,133,
    150,56:REM 2156
D8  180 DATA 87,237,64,31,133,73
    ,165,90,237,65,31,5,73,144,1
    0,165:REM 1620
2C  190 DATA 87,141,64,31,165,89
    ,141,65,31,160,87,133,85,165
    ,90,133:REM 1715
A0  200 DATA 86,165,85,208,2,150
    ,86,150,85,165,83,157,87,208
    ,6,165:REM 2084
2C  210 DATA 84,157,90,240,25,16
    0,0,177,95,145,87,165,85,208
    ,2,150:REM 1550
```

```
94  220 DATA 96,198,96,165 87 20
    0 2 198 88 198 87 76,31 49 1
    65,60 1653
A1  230 DATA 133,91 32 106 45 24
    ,165,160 169 57,31,133,188,1
    44 2 230 1718
2E  240 DATA 169,230 00,165 00 1
    97,85 206,229 96,127 127 127
    ,127,127 127,2302
2C  250 DATA 127,127,127,127,165
    ,167,133 169,133 00,133,90,5
    6,165 166,229,2282
74  260 DATA 172,133,168 133,89
    176,4 198,169,100,90,24,165
    168 189 57,2053
07  270 DATA 31 133,07,144,2,200
    ,00,50 165 07 237,64 31,133
    91,165 1744
DE  280 DATA 00,237,65 31 5,91,1
    44 3,76 120 31,160,0 177,07,
    145 1450
E8  290 DATA 05 200 05,200 2,200
    ,50 230 07 200 2,230 00,173
    64,31 2051
7A  300 DATA 157 07,200,231,173,
    65 31,157 00 200,224,100,0,1
    65 07,197,2310
3E  310 DATA 05,200 6,195 00,197
    50,240 14,165 07 200 2 190,
    00,190,2043
83  320 DATA 07,165 32,145,57 20
    0 230,105,171 133,91 32,100
    45,165 85,2023
04  330 DATA 141,54,31 165 90 14
    1,65 31,70,120,31,253 255 25
    5,255,255 2220
E5  340 DATA 255,255 255,255,255
    ,255 255 255,255 255,221,46,
    46 46 46 50 3013
2F  350 DATA 46,46 46 46,221,48,
    46,46 46,50,46 48 46,46,221,
    48,1050
EE  360 DATA 46,46,45 50,48,48,4
    6,45 221,46 46,46,46,90 48 4
    6,935
57  370 DATA 48 48,221 45,46 46,
    46 50 46 46 40 40,221,46 46,
    46 1050
32  380 DATA 45,55,46 46,46 46 2
    21 46,48 48 48,50 46,46 46 4
    6 935
9E  390 DATA 221,48,46 46,48 55,
    46 46,46,46,0,127,32,50 48 1
    59,1075
2F  400 DATA 120,141 167 2 169,2
    26 141,168,2,169 90,141,169,
    2 169 234 2070
99  410 DATA 141 170 2 169 0 141
    171 2,162 0 189 0 50 201 46
    240,1664
78  420 DATA 27,201 47,240 23,16
    9,46 157 0 50 134,74,32 72 4
    0 166 1470
20  430 DATA 74 232,224 00 200 2
    39,32 63,40 76 120,31 162 79
    ,189 19,1860
04  440 DATA 39,157,0 50,202 16,
    247 162,0,189 0 50 134 73,32
    72 1423
87  450 DATA 40,190 73,232,224 0
    0 200,241,76 140,50,255,255
    255 255 255 2805
8C  460 DATA 255 255,255 255,255
    ,255 255,255 255 255,255,253
    255,255 0 40 3616
D0  470 DATA 0 40 76 200 50,76 5
    51,56 165 166,237,64,31 133
    73 1437
50  480 DATA 165,167,237,65 31 5
    73,176,2,144,31 165,190,200
    ,2 190,1835
F3  490 DATA 167,190 166,160,0,1
    77 166,201 32 240 5,230 173
    200,2 230 2359
44  500 DATA 174 32 5,51 200 190
    200,2 230 167,169,32 76,254
    41,32 1869
1F  510 DATA 90 40,165 112 141 1
    67 2 169 224,141,189,2,169,1
    3 141 169 1007
78  520 DATA 2,169 204,141,170 2
    ,169 1,141 171 2,166,173,165
    ,174,32,1002
00  530 DATA 255,105,165,174 200
    ,22 165,0 0 0 0 0,0,0,0,0,11
    20
```

PROGRAM YC WRITER L

```
2C  10 DL=50 LN=90 SA=1310
    1
50  20 FOR L=0 TO DL CX=0 FOR D=
    0 TO 15 READ A CX=CX+A POKE
    SA+L*15+D,A NEXT D
A5  30 READ A IF A><CX THENPRINT
    "ERROR IN LINE",LN+(L*10) ST
    OP
40  40 NEXT L END
00  50 DATA 173,201,10,176 7 169
    ,32 32,210,255 200 6 201,100
    ,176,5,1981
20  60 DATA 169 32 32 210,255 32
    ,93 40,90,77,78,90,69,32,84,
    65,1421
EE  70 DATA 66 00 79,83 73,84 73
    ,79 70,32,70 79,82,87,65,82,
    1152
90  80 DATA 00,169,0,133 173,133
    ,174 165 166 133 87 165,167,
    133 88,200 2184
0F  90 DATA 87,255 2,230 90,165,
    87,205,64,31,200,0 165,90,20
    5,65,1945
F4  100 DATA 31,200,1,90 120,173
    ,69,31 133 87 133,52,173,70,
    31,133 1581
EA  110 DATA 88,133,53 24,105,52
    ,109 88 31,133,52 144,2,200
    93,165 1650
C0  120 DATA 0 133,85,133,50,133
    ,51,160 0 177,87,201 32 240,
    0 169 1741
63  130 DATA 0,133 89 240,14 165
    89 200,10 169,1,133,89 200,
    173 200 1551
47  140 DATA 2 200,174,230,87,20
    9,2 200 00 165,87,205,64,31,
    200 215,2226
3C  150 DATA 165,88,205 65,31 20
    0 200 50,173 64 31 229,52,13
    3,94 173 2015
50  160 DATA 65 31,225,53,5,94,1
    44 35 160 0 177,52,201,32,24
    0,13 1611
01  170 DATA 200,177,52,201,32,2
    40 5 230 50,200 2,230,51,24,
    165 52,2000
00  180 DATA 105 57,31,133,52,14
    4 200,230,53 176,204 24,165,
    173 101,90 2030
50  190 DATA 133 173 165 174 101
    ,91,133,174 00 32 25 52 32 2
    03 50 96,1785
7C  200 DATA 32 60 40,169,112 14
    1 167 2 169 224,141 168 2 16
    9 13,141 1750
85  210 DATA 169,2 169 204 141,1
    70 2 169 1,141 171 2 162 5,1
    65 32,1709
E2  220 DATA 32,210,255,202 206
    250 32 63 40 96,76,77 52 76
    132,53,1954
60  230 DATA 32 123 31,32,92 193
    ,169 60 133 178 169 3 133 17
    9 162 15,1784
E0  240 DATA 165 87 31 261 32 20
    0 3 202 208 246 232 130 162
    97 160 31 2237
90  250 DATA 32 169,255 169 1 17
    4 74 31,160 32,100,255 169,0
    32 213 1580
45  260 DATA 255 144,13,173 74 3
    1,201 1 240,3,32,170 54 76 1
    17 31 1615
90  270 DATA 142 64 31 140 65,31
    169,0 133,190 173 65,31 133
    ,190 133 1666
20  280 DATA 166 173,76 31 133,1
    65 133 167 133 252 56,165 16
    6 233 7 133,2107
27  290 DATA 251 176 2,198 252 1
    60 0 177 251,141,57 31 200 1
    77 251 141 2465
1E  300 DATA 58 31 200 177,251 1
    41 59 31 200 177,251 141 60
    31 200 177 2165
47  310 DATA 251,141,61,31 200,1
    77 251 141,62 31 200 177,251
    141 63 31 2205
20  320 DATA 32 61 40,173 59 31
    141 167 2 173 60 31,141,166,
    2 173,1474
DA  330 DATA 61 31,141,169 2,173
    62 31,141 178,2 169 0,141 1
    71,2 1466
CE  340 DATA 32 100,45,173,03 31
    200 13 173 57,31,201,00 240
    ,60 32 1627
90  350 DATA 60 45,76,41 53 32,5
    0 40 173,03,31,74 133 73 32,
    100 1169
C9  000 DATA 20,24 173 63 31,109
    ,57 31 201 00 240 36 173 50
    31 74 1409
6E  370 DATA 170 24 173,167 2 15
    5 0 141 167 2 144,3 238 100,
    2 202 1716
F5  380 DATA 200,235 165 172,72
    173 50 31 133 172 32 191 20
    104 133 172,2003
07  390 DATA 32 63 40 120,32 135
    194 32 236 44 32 235 44 32,
    94 51 1420
0C  400 DATA 76 120 31 10 32 80
    02 69,03 03 32 65 70 85 32,7
    5 1045
C1  410 DATA 85,69,32 84,79,32,0
    2,65 84,85,82 70 32,84,79,32
    ,1092
16  420 DATA 84 69 80 84 32,146
    0,32 123 31,32,60 228 24,162
    12 1216
1F  430 DATA 160 0 32,240 255 17
    3,74 31 201 1 240,10 165,255
    240 6 2091
0C  440 DATA 32 0 62 76 170 53,1
    65 80 160 54 32 30 171 169,9
    ,133,1400
```

```
E5  450 DATA 211,189,185,158,54,
    32,38,171,32,228,255,240,251
    ,201,78,208,2426
87  460 DATA 3,76,77,54,201,89,2
    00,240,173,74,31,201,1,240,7
    ,185,1848
94  470 DATA 255,240,3,32,11,62,
    173,78,31,133,252,58,173,69,
    31,233,1824
58  480 DATA 7,133,251,176,2,158
    ,252,168,0,173,57,31,145,251
    ,200,173,2289
AF  490 DATA 58,31,145,251,250,1
    73,58,31,145,251,200,173,00,
    31,145,251,2284
91  500 DATA 200,173,01,31,145,2
    51,200,173,02,31,145,251,200
    ,173,03,31,2138
11  510 DATA 145,251,189,80,133,
    170,169,3,133,178,102,16,202
    ,189,48,31,2100
0F  520 DATA 201,32,240,240,232,
    138,162,00,189,31,32,189,255
    ,159,3,174,2340
58  530 DATA 74,31,189,255,30,10
    0,255,159,251,174,64,31,172,
    65,31,232,2102
79  540 DATA 200,1,200,32,221,24
    5,173,74,31,201,1,240,3,32,1
    70,54,1006
B9  550 DATA 0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0

 PROGRAM  YC WRITER M

14  10 BL=50     LN=50     SA=1390
    1
59  20 FOR L=0 TO BL:CX=0 FOR D=
    0 TO 15 READ A CX=CX+A POKE
    SA+L*16+D,A NEXT D
A5  30 READ A IF A><CX THENPRINT
    "ERROR IN LINE";LN+(L*10) ST
    OP
40  40 NEXT L END
D4  50 DATA 78,117,31,82,85,65,6
    8,59,38,84,79,32,83,65 58,80
    ,1127
5D  60 DATA 32,84,69,80,84,78,73
    ,78,69,33,13,13,0,10,32,65,0
    ,19
3D  70 DATA 82,69,32,83,79,89,32
    ,83,69,82,69,83,32,48,09,47
    ,1050
99  80 DATA 78,41,32,140,13,13,0
    ,185,0,32,185,255,185,15,182
    ,8,1322
1E  90 DATA 100,32,196,255,32,19
    2,255,162,15,32,150,255,158,
    144,206,3,2305
D3  100 DATA 32,207,255,32,210,2
    55,76,153,54,32,204,255,96,1
    69,13,32,2075
F3  110 DATA 210,255,32,210,255,
    169,7,133,211,169,1,133,204,
    32,132,54,2207
49  120 DATA 165,13,32,210,255,3
    2,210,255,158,4,133,211,189,
    96,160,53,2171
D9  130 DATA 32,30,171,32,228,25
    5,240,251,96,169,13,32,210,2
    55,32,210,2256

FB  140 DATA 255,189,4,133,211,1
    58,110,158,53,32,58,171,32,2
    28,255,240,2252
F8  150 DATA 251,96,81,173,33,20
    8,133,82,173,32,68,229,32,15
    2,195,32,1570
48  160 DATA 182,195,173,32,208,
    133,81,173,33,208,133,82,173
    ,134,2,133,2075
CD  170 DATA 83,158,0,141,32,208
    ,141,33,208,169,1,133,00,32,
    170,55,1655
B6  180 DATA 32,220,255,240,251,
    201,13,208,24,165,80,208,10
    ,165,1,133,2210
07  190 DATA 00,32,170,55,76,29,
    55,169,0,133,00,32,147,58,76
    ,29,1221
0F  250 DATA 95,201,133,208,219,
    165,81,141,32,208,165,82,141
    ,33,208,196,2237
EF  210 DATA 83,141,134,2,120,32
    ,132,104,70,120,31,32,58,229
    ,70,120,1558
41  220 DATA 31,7,171,58,87,79,7
    6,79,65,82,0,158,7,154,83,75
    ,1184
F3  230 DATA 73,80,53,0,110,7,17
    8,52,87,79,76,79,86,82,0,120
    ,1168
F7  240 DATA 7,58,0,158,7,83,75,
    73,80,53,100,35,58,84,89,58,
    1079
DF  250 DATA 84,58,0,148,7,177,3
    5,62,84,58,80,84,58,0,158,7,
    1103
EF  258 DATA 153,30,65,58,48,65,
    0,100,7,59,0,178,7,76,170,55
    ,1167
E1  270 DATA 76,61,56,76,80,58,1
    05,0,133,87,169,4,133,00,169
    ,0,1503
93  258 DATA 133,89,158,210,133,
    90,32,90,229,158,0,169,85,14
    5,87,169,1574
A1  290 DATA 0,145,89,200,169,57
    ,145,87,169,0,145,89,192,30,
    200,243,1958
48  300 DATA 200,158,73,145,87,1
    85,6,145,89,32,01,58,105,156
    ,58,240,1063
30  310 DATA 9,145,87,189,0,145
    89,200,200,242,32,01,58,106,
    151,56,1001
76  320 DATA 240,9,145,87,189,0,
    145,89,200,200,242,32,01,58,
    105,74,1532
1E  330 DATA 145,87,169,0,145,89
    ,200,105,87,145,87,185,6,145
    ,89,132,1510
F2  340 DATA 58,208,243,200,105,
    75,145,87,189,0,145,89,32,01
    ,58,32,1755
13  350 DATA 61,58,169,232,133,0
    4,169,58,133,89,162,10,32,06
    ,56,96,1626
DE  360 DATA 24,165,87,105,40,13
    3,87,144,2,250,80,24,165,89,
    105,40,1529
58  370 DATA 133,89,144,2,230,90
    ,160,0,58,169,0,133,73,160,0
    ,177,1656
23  380 DATA 84,240,36,201,254,2
    40,15,201,255,208,17,32,61,5
    6,255,84,2214
5D  390 DATA 208,235,230,85,208,
    231,169,128,133,73,208,242,6
    9,73,145,87,2524

28  450 DATA 185,6,145,83,200,20
    0,216,24,200,152,101,84,133
    ,84,144,2,1557
95  410 DATA 230,85,32,81,58,202
    ,208,147,96,83,32,32,32,32,3
    2,32,1452
58  420 DATA 32,38,32,42,42,42,3
    2,39,25,3,32,23,10,9,20,5,42
    ,0
C5  430 DATA 10,38,32,42,42,42,3
    2,32,32,32,32,32,32,32,32,32
    ,535
83  440 DATA 93,0,93,40,3,41,32
    49,57,58,55,32,2,45,0,32,830
58  450 DATA 12,5,0,13,1,14,14,3
    2,32,32,32,32,22,5,10,19,201
C0  460 DATA 9,15,14,32,1,47,53
    47,48,53,0,20,5,24,20,45,474
C4  470 DATA 5,14,20,10,25,32,13
    15,4,5,19,58,0,45,45,45,303
57  480 DATA 45,45,45,45,45,45,4
    5,45,45,45,45,45,45,0,23,23
    053
A9  490 DATA 32,45,32,23,15,10,4
    ,32,23,10,1,16,32,15,14,47,3
    07
BC  500 DATA 15,6,6,0,10,32,45,3
    2,10,9,7,0,20,32,10,21,271
C5  510 DATA 15,20,5,0,5,3,1,20
    ,9,15,14,32,15,14,47,15,240
04  520 DATA 6,6,0,9,32,45,32,9,
    14,15,5,10,20,32,13,15,275
A0  530 DATA 4,5,32,15,14,47,15,
    6,0,0,255,0,15,10,13,1,452
63  540 DATA 20,9,14,7,32,0,1,3,
    9,12,9,20,9,5,19,58,233
99  550 DATA 0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0

 PROGRAM  YC WRITER N

15  10 BL=58     LN=58     SA=1470
    1
58  20 FOR L=0 TO BL CX=0 FOR D=
    0 TO 15 READ A CX=CX+A POKE
    SA+L*16+D A NEXT D
A5  30 READ A IF A><CX THENPRINT
    "ERROR IN LINE" LN+(L*10) ST
    OP
40  40 NEXT L END
E5  50 DATA 0,45,45,45,45,45,45,
    45,45,45,45,45,45,45,45,45,6
    75
20  58 DATA 45,45,45,45,45,45,0,
    3,32,45,32,21,14,45,10,21,49
    3
0C  70 DATA 19,20,9,5,25,32,16,1
    10,1,7,10,1,16,0,32,229
C7  80 DATA 21,14,4,5,10,32,3,21
    ,10,15,15,10,0,4,32,45,269
8E  96 DATA 32,10,3,7,0,20,32,10
    ,21,19,20,9,5,25,32,15,204
9A  100 DATA 1,10,1,7,10,1,16,0
    ,32,21,14,4,5,10,32,3,199
EC  110 DATA 21,10,19,15,10,0,1
    ,32,45,32,9,14,19,5,10,20,286
```

```
30  120 DATA 32,5,13,16,20,25,32
    ,12,5,14,5,32,1,20,32,3,271
0C  130 DATA 21,18,19,15,18,0,25
    5,5,18,1,19,5,32,6,1,3,436
20  140 DATA 9,12,9,20,5,5,19,58
    ,0,45,45,45,45,45,45,45,456
FE  150 DATA 45,45,45,45,45,45,4
    5,45,45,45,0,2,32,45,32,5,56
    6
DC  160 DATA 18,1,19,5,32,12,9,1
    4,5,32,21,14,4,5,18,32,241
D0  170 DATA 3,21,18,19,15,10,0,
    14,32,45,32,5,10,1,18,5,282
EC  180 DATA 32,3,21,18,18,5,14,
    20,32,2,18,15,3,11,0,5,211
0C  190 DATA 32,45,32,5,18,1,16,
    5,32,23,0,15,12,5,32,20,304
64  200 DATA 5,24,20,6,6,12,5,0,
    255,255,254,32,32,6,46,32,55
    5
66  210 DATA 45,32,2,1,3,11,32,2
    0,15,32,20,5,24,20,32,32,326
DE  220 DATA 32,10,5,20,32,45,32
    ,13,15,10,5,32,5,5,12,16,300
2A  230 DATA 32,32,32,0,66,0,169
    ,0,133,07,166,4,133,08,166,0
    ,1137
6F  240 DATA 133,06,166,210,133,
    08,32,68,225,165,100,133,04,
    169,58,133,2085
F7  250 DATA 05,162,21,32,176,66
    ,66,2,12,16,3,11,45,13,15,22
    ,765
D6  260 DATA 6,32,6,1,3,5,12,9,2
    0,6,5,16,30,0,45,45,278
FD  270 DATA 45,45,45,45,45,45,4
    5,45,45,46,46,45,45,45,45,45
    ,720
FC  280 DATA 45,46,46,45,0,7,32,
    45,32,16,5,20,32,16,20,1,412
6F  290 DATA 18,20,47,6,14,4,32,
    15,6,32,2,12,15,3,11,0,230
67  300 DATA 0,32,45,32,3,15,16,
    25,32,3,21,10,10,5,14,20,307
91  310 DATA 32,2,12,15,3,11,32,
    20,15,32,3,21,10,16,15,10,20
    0
CC  320 DATA 0,15,32,45,32,13,15
    ,22,5,32,3,21,10,10,6,14,290
2D  330 DATA 09,32,2,12,16,3,11,
    32,20,15,32,3,21,10,16,15,27
    0
C0  340 DATA 10,0,206,13,1,10,7,
    6,14,32,0,1,3,6,12,6,407
99  350 DATA 20,6,5,16,30,0,45,4
    5,45,45,45,45,46,45,45,45,30
    1
FD  360 DATA 46,45,45,45,45,45,4
    5,45,0,24,32,45,32,16,5,20,5
    37
E4  370 DATA 32,12,5,0,20,32,13,
    1,16,7,6,14,0,25,32,45,271
FE  380 DATA 32,19,5,20,32,10,5,
    7,8,20,32,13-1,10,7,9,250
C2  390 DATA 14,0,255,20,1,2,32,
    6,1,3,6,12,6,20,6,5,330
3C  400 DATA 16,56,0,45,45,45,45
    ,45,45,45,45,45,45,45,45,45,
    802
D2  410 DATA 45,45,0,6,53,32,45,
    32,10,21,13,16,32,20,15,32,4
    17
64  420 DATA 14,5,24,20,32,20,1,
    2,45,16,15,19,5,20,5,15,265
C3  430 DATA 14,0,6,54,32,45,32,
    10,21,13,16,32,20,15,32,6,34
    0
F1  440 DATA 15,10,13,5,10,32,20
    ,1,2,45,16,15,19,5,20,5,257
4D  450 DATA 15,14,0,6,32,45,32,
    3,12,5,1,10,47,10,5,13,272
4B  460 DATA 20,15,10,5,32,1,12,
    12,32,20,1,2,45,16,15,13,265
EA  470 DATA 5,20,5,15,14,15,0,2
    0,32,45,32,3,12,5,1,10,254
F1  480 DATA 47,10,5,13,5,15,10
    ,5,32,13,6,14,7,12,5,32,277
12  490 DATA 20,1,2,45,16,15,16,
    9,20,5,15,14,0,205,16,20 475
A4  500 DATA 0,6,10,19,60,0,46,4
    5,46,45,45,45,46,32,32,32,51
    9
9A  510 DATA 32,32,32,32,32,32,3
    2,32,32,63,32,21,32,45,32,23
    ,566
4F  520 DATA 15,10,4,3,15,21,14,
    20,0,12,32,45,32,12,15,1 258
04  530 DATA 4,32,6,6,12,6,32,32
    ,32,32,32,83,32,14,32,431
F7  540 DATA 46,32,3,0,1,14,7,6,
    32,6,5,12,5,14,1,13,207
00  550 DATA 0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,0

PROGRAM  YC WRITER O

70  10 GL=36      LN=50     SA=1550
    1
58  20 FOR L=0 TO GL CX=0 FOR D=
    0 TO 15 READ A CX=CX+A POKE
    SA+L*16+D,A NEXT D
A5  30 READ A IF A>CX THENPRINT
    "ERROR IN LINE";LN=(L*10) ST
    OP
AC  40 NEXT L:PRINT"{DOWN2}GET R
    EADY TO SAVE AND PRESS A KEY
    " POKE136,0 WAIT180,1
ED  46 POKE43,59:POKE44,22 POKE4
    5,142 POKE46,62 X=PEEK(186)
    SAVE"WRITER2" X
2F  50 DATA 5,0,15,32,45,32,15,1
    ,22,5,32,6,6,12,5,32,270
EE  60 DATA 32,32,32,32,32,32,63,32
    ,13,32,45,32,3,0,1,14,7 440
7C  70 DATA 5,32,3,15,12,15,21,1
    4,5,0,10,32,45,32,10,5,260
1F  80 DATA 16,12,1,3,5,32,6,9,1
    2,5,32,32,32,63,32,16,330
60  36 DATA 32,45,32,16,16,6,14,
    20,15,21,20,0,255,254,32,32
    015
02  160 DATA 6,46,32,45,32,2,1,3
    ,11,32,20,15,32,20,5,24,329
50  110 DATA 20,32,32,32,10,5,20
    ,32,45,32,13,15,10,5,32,0,35
    5
0A  120 DATA 5,12,16,32,32,32,0,
    32,123,31,32,68,229,24,162,1
    0,648
4E  130 DATA 160,10,32,240,205,5
    6,173,32,208,233,240,133,77,
    166,104,160,2282
C7  140 DATA 61,32,169,61,144,3,
    141,32,208,56,173,33,208,233
    ,240,133,1527
51  150 DATA 77,169,120,160,61,3
    2,169,51,144,3,141,33,208,17
    3,134,2,1687
A3  160 DATA 133,77,169,135,160,
    61,32,169,61,144,3,141,134,2
    ,198,211,1030
1E  170 DATA 169,140,160,61,32,3
    0,171,32,220,255,240,251,201
    ,70,240,170,2466
7D  180 DATA 201,89,208,243,120,
    32,132,154,76,120,31,76,79,0
    2,60,65,1010
AC  190 DATA 84,32,61,76,76,76,0
    5,02,30,32,0,80,65,80,66,82,
    1240
64  200 DATA 32,67,76,76,76,05,0
    2,60,32,0,73,78,75,32,67,76,
    994
0A  210 DATA 76,76,05,02,50,32,0
    ,10,32,03,65,04,73,03,78,73,
    663
5A  220 DATA 66,60,63,32,40,89,4
    7,78,41,32,146,0,32,30,171,1
    65,1103
C3  230 DATA 211,72,166,0,166,77
    ,32,205,106,104,133,211,162,
    0,134,73,1630
AD  240 DATA 134,74,134,75,32,22
    0,255,240,251,201,13,240,20
    ,201,48,144,2296
35  250 DATA 213,201,58,176,238,
    32,216,255,73,48,160,75,146,
    73,232,224,2454
3E  260 DATA 2,134,75,240,2,200,
    221,166,13,32,210,256,32,210
    ,255,106,2227
09  270 DATA 10,133,211,106,75,2
    00,2,24,56,24,162,10,165,0,1
    01,73,1464
5E  280 DATA 202,208,251,101,74,
    56,66,55,42,42,42,76,14,62,7
    6,22,1423
22  290 DATA 62,166,88,100,62,32
    ,30,171,66,102,0,100,70,31,1
    57,65,1552
3D  300 DATA 3,201,32,208,11,232
    ,106,70,31,201,32,200,230,20
    2,260,5,2075
04  310 DATA 232,224,10,208,230,
    166,0,157,05,3,169,0,32,105,
    205,166,2120
0E  320 DATA 16,174,74,31,160,32
    ,196,255,32,162,256,162,15,3
    2,201,205,2076
1A  330 DATA 166,65,100,3,32,30,
    171,32,101,171,60,02,66,65,0
    0,09,1403
80  340 DATA 32,84,76,32,02,65,0
    0,76,65,67,66,32,84,65,60,84
    ,1852
27  350 DATA 70,73,76,66,33,13,1
    3,0,0,0,0,0,0,0,0,0,347

PROGRAM  WRITER BOOT

C9  5 A=A+1 Y=PEEK(186)
13  10 IFA=1THENLOAD"WRITER1",Y,
    1
B9  20 IFA=2THENLOAD"WRITER2",Y,
    1
43  30 IFA=3THENSYS50050
```

Technical Information

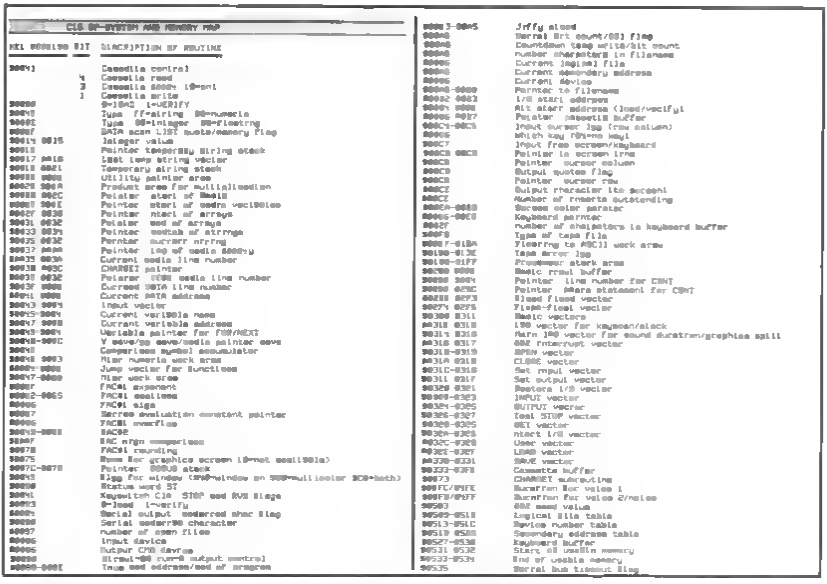
All you ever wanted to know about your Commodore but were afraid to ask.

Most programmers spend a lot of their time sifting through piles of technical books looking for the address of a certain routine or trying to find the POKE to perform a certain function.

Now you can throw away your books, as on the following pages you will find a wealth of information about all of the popular Commodore computers.

Advanced programmers will find the memory maps invaluable while both beginners and old hands alike will find the Hex converter, the hints and tips and much more to their liking.

Most of the information provided here is useful by itself. Some information, such as the addresses of routines within the ROMs, will be of more use when used together with a ROM disassembly.



```
$0030          Current color code
               7  1=Blink
               4-6  Luminance (0 TO 7)
               3-0  Color (0-15)
$A538          Max size of keyboard buffer
$D041          Repeat speed counter
$D042          Repeat Delay counter
$D043          Keyboard shift/control flag
$D045          Case switch sound
$D045-D046     Keyboard table setup pointer
$D057          Case switch disable
$D780-D7FF     Basic track
$D07B          SYS A reg save
$D07A          SYS X reg save
$D079          SYS Y reg save
$D077          SYS return reg save
$D078          Last key
$D07E          Countdown for double TI bump
$D800-DBFF     Color memory
$DC00-DFFF     Screen memory
$1000-3FFF     Basic program memory
$1000-3FFF     Screen/color memory
$3FF8-3FFF     Reset entry (when 000 in ROM)

Memory map

$A000          Basic ROM cold start routine
$E0A3          Initialize basic
$B0CC          Power up message
$0300          Vector for $B300
$A117          Initialize vectors
$E3A3          CBASE7 for $E57B
$A1AE          Keyword
$A393          Command vectors
$A19B          Evaluate vectors
$A003          Arithmetic operator vectors
$A47A          Error messages
$A000          SYNTAX ERROR
$A943          Error routine
$A763          READY
$A74F          Basic warm start
$A53B          Find basic line
$A67A          Perform NEW
$A68E          Perform CLR
$A68F          Get text pointer
$A6AF          Perform LIST
$A6BC          Perform RUN
$A7AE          Perform EXECUTE
$A8CB          Perform STOP
$A8D0          Perform END
$A8E4          Perform CONT
$A8F8          Perform DEL
$A906          Perform PRINT#
$A912          Perform PRINT
$A9BB          Print string from ly ri
$A9D0          Perform GET
$A9E7          Perform INPUT#
$A9E0          Perform INPUT
$AA1F          Perform READ
$A000          "READ ERROR START" message
$AA80          Perform NEXT
$AAEC          Evaluate expression
$AA36          Constants fi
$AB4E          Perform NOT
$AA68          Scan parm )
$AB6C          Comr pass 1
$AB91          Setr pass
$AB7B          Perform OR
$AB7A          Perform AND
$AB9B          Perform comparisons (<=>)
$AB3B          Perform DIM
$A000          Locate variable
$B08C          Convert   367BA
$A000          Find-Make
$B4AC          Perform ABS
$B4CF          Perform POS
$B4D4          Perform SGN
$B4E0          Perform INT
$B4B4          Perform STR$
$B4C4          Get keyperry rString
$B5EB          Perform CHR$
$B5CF          Perform LEFT$
$B603          Perform RIGHT$
$B615          Perform Mid$
$B67A          Perform LEN
$B67B          Perform ASC
$B6A4          GOM i byte parameter
$B6D3          Perform VAL
$B6DE          Get GRCombters for POKE/wait
$B6EA          Perform PEEK
$B6FB          Perform POKE
$B6E0A          Perform WAIT
$B6E7          Perform FLOAT#C1
$B6EE          Perform ADD
$B6E1E          Perform LOG
$B478          Perform MULTIPLY
$B4A7          Perform DIV(10)
$A6E1          Memory Pr FAC#1
$A6A0          CmpErr for FAC#1
$A6AB          Round FAC#1
$A6BC          ReStore FAC
$A6E8          Perform ABS
$A35B          Perform INT
$A5E4          Perform SGN
$A5EB          Perform FUNER
$B6B7          Perform NEGATE
$B000          Perform EXP
$A7A7          Perform RND
$A773          Basic I/E error handler
$A7B5          Basic kernal patch for OPEN
$A7Bb          Basic kernal patch for PRINT
$A7E1          Basic kernal patch for INPUT

$A7E7          Basic-kernal patch for GET output device
$A7A5          Getic-kernal patch for GET input device
$A7AF          Getic-kernal patch for GET
$A7B5          Perform SYS
$A7BE          Perform SAVE
$A7F0          Perform VERIFY
$A7F3          Perform LOAD
$B000          Perform OPEN
$B000          Perform CLOSE
$A000          GET parameters for LOAD/SAVE
$B000          GET parameters for OPEN/CLOSE
$B000          Garbage collect
$A47A          Perform OOG
$B877          Perform SIN
$B8CD          Perform TAN
$B81A          Perform ATN
$B000          Perform REAL#PEEK
$B4CA          Perform POW
$A25A          Perform SELECT
$A4CA          Bad range for LIST/DELETE
$B4E0          Perform ?PWR
$B4A0          Perform MERGE
$B000          Perform PUWRF
$B8B7          Perform RN
$B8AC          Perform EXII
$B8E3          Perform LOOP
$B000          Perform TRON
$B000          Perform TROFF
$B8CB          Perform AUTO
$B000          Perform XELF
$B7E8          Perform RET
$B000          Perform $BUMB
$B000          Perform UBL
$B001          Perform PRINT
$B000          Perform CHAR
$B000          Perform Box
$B000          Perform BORAFE
$B2E8          Perform BORAFE
$CB1E          Perform CIRCLE
$C300          Perform GRAW
$C3BF          Perform LOCATE
$C3Ir          Perform COLOR
$C3E7          Perform SCNCLR
$C3EB          Perform SCALE
$C3C3          Perform GRAPHIC
$C3EC          Perform DISACTRON
$CPb1          Perform BRAW?
$CB51          Perform RLOAD
$C3BF          Perform MEARUEN
$C3BC          Perform SCRATCH
$C3CC          Perform COLLEC?
$C3BW          Perform DOPY
$C3F1          Perform RENAME
$C3BF          Perform BACKUP
$CBE1          Set parameters for disk commands
$CCCF          Read disk error
$C3BF          ARE YOU SURE  message
$C7E0          IBO BAK relay
$C3BW          IBO readline($0B1)-terminm graphics rplit sound SwrRirn
$C3E2          IBO readline($0101) handles block keyberrs
$C3C3          Scroll sound durrl.ong
$C1F0          Bump clock
$C7B0          Set time
$C7F0          Bar time
$D000          Character define.tens LCK1
$D000          Screen memmas LiDal
$D01A          Cursor address (X!B)-1
$D02A          Dal screen rise
$D000          Put/Get Row Col
$D000          Set screen pointer
$DBC1          Remove char from keyboard
$D000          Input until carriage return
$D011          Read keyboard
$BC90          Output to screen
$D011          Keyboard select vectmr
$E5E6          Unshift$Mm Table
$ED67          Shifted table
$D000          CBM key table
$D000          Control table
$D1B4          $0112/Trn equivalent
$E2EA          Delay for 8 @ one
$E3BC          Find ary temp memdac
$EAA1          Find r specific memdac
$D000          Out (.00+20)
$D000          Input (.00320)
$BC90          Output (.00$0+1
$BC80          Remove from corri
$BC8F          Read corri deferred
$D01B          Set input device ($03IC)
$D000          Set output device (.003IE)
$E0DA          Send TALK
$B91A          Read TALK Ba
$BEBC          Send LISTEN
$B97B          Read LISTEN Ba
$D000          Close file (.003Ir)
$BLEE          Find file entry
$E2F0          Get file details
$AF80          Open# rli file (.003E+)
$F0BC          Restore default I/E  $0000:
$BAE3          Send UNLISTEN
$B7B0          Send UNTALK
$BF53          Open file (.003LE)
$F09A          Load program (.003IE)
$F1A4          Save program (.00330)
$D000          Test STOP key (.00300)
$F2A4          Power reset entry
$F2CC          Get kernal vectors
$F2ED          Store kernal vectors
$F300          Initialize I/O
$F32C          Initialize system constants
$F34C          Get Bl$rmma Datalr
$F413          Set Rlr detailr
$F41A          Blmg status
$F41C          Get status
$F423          Set Bl$Rmma
$F42D          Read/set top of memory
$F42F          Send top of memory
$F42F          Set top of memory
$F436          Send/set bottom of memory
$F43B          Read Bottom of memory
$F43E          Set bottom of memory
```

```
$fff5          Monitor call entry
$fffC          Monitor BRK entry (00310) user vector $0302
$fc13          Get I/O address
$fc33          100 entry
$fc3E          100 exit
$fc9A          I10 opevtr
$ff52          Fortran MONITOR
```

6502 MEMORY MAP

SEC #LOCES	DESCRIPTION
0	7501 bstr Direction registor
1	7501 0 tit I/O port inaCS41
f-4	Now rtart address: I6E%n%QE1
5-8	Stop rith (D2%n%881
7	Search rharacter
8	Flag searching for quote
t	abrean rrilen tree last TAB
10	tlag 0=LOAD 1=VLfif7
11	input buffer rointer number of rilenoer
13	tlr $FF=string $00=numerir
14	tlag $ff=integer nonstlaatling
15	flag abr mean LIST quotr/nonmay flag
16	Flag user tunction call
17	tlag $80=INPut $40=BET $08=READ
18-21	Integer value
22	Pointer temporary string rtack
23-24	Vector trrt temporrry string
25 26	Temporary string rtack
2A 37	Utility pointrr area
38 48	Product area tor multiplication
49-4A	Pointer rtrt of addir
4B-4D	Pointer rtart of addir variables
4T 4E	Pointrr rtart of addir arrar
4F-50	Pointrr rPe of arrays
51-52	Frimtrr rtart of strings
53-54	Pointer rurrent string
55-58	Pointrr tpe of basir memory
07 58	Current basir line number
5A-5B	Previous basic line number
01-62	Pointer CONT addir line number
63-64	Current DATA line number
65-66	Pointer current DATA address
67 68	Vector input routine
58 70	Current variable name
71-72	Variable address
7F 74	Vacirtir pointer for FOR/NEXT
75-76	7 save rt save addir pointer save
77	Comparison 1=larger 2=equal t=smaller
78-8f	Misc eprt rtne
8n-88	Vector functions
07 88	Risk eprt Acsh
87	Accumulirtor FI exponent
88-101	Accumulirtor FI mantissa
100	Accumulirtor 81 rlag
163	Series evrluation constant counter
104	Accumulirtor %1 tl order overflow
105 110	Accumulirtor 0E rr fnc rl
111	Accumulirtor rign comparison
113	Accumulirtor FI rounding
113-114	Pointer rassette buffer
110 110	flrg AUTO command $00=0FF
117	tlag 0 - 101 reserved tor graphics
10n-108	Pointrr $0000 Stack
131	Currrer grraphics mode mnemonr $0000=low
104	$00=Wait hirgs
130	$00=multinolar $00=multinolar $01=t
1FF	Current color
124	Multi rrlor 1
125	Foreground color
135	Mar number of columns
136	Max number of rows
144	Strtus word ST
145	tlag $10F rrs $40 key
147	tlag mnemonr $00=VERIFY
148	tlr nPerrtor in oemirl buffer $00=no $80=YES
149	Charrrrrt ir buffer for oemirl address
181	Number of tiles open
152	abtault input device
153	Devirr output device
154	flag $00=direct mode $C0=emitter $00=progress
177 108	Pointrr tape end/progress end
18f-100	Jiffy timer
171	tength rf filename
17t	tagirrl tlir number
173	Secondary address
174	devirr number
175-178	Pointer filename
179-175	I/O Prrt address
188-181	abrir loading address
182-183	Pointer load end address for tape
184	Firt RVS $12=Yes $00=No
185-187	Current emaitter ix yi
188	tlrg key pressed $0/=none
189	Input from screen/keyboard
200-201	Pointrr screen line
202	Pointer screen colum
207	flrg $00=npr in quott mode
209	Length of current screen line
205	Pointer cursor rpa
206	Output rharrcter tr screen
207	tlag insert mode 1 $00=number of inserts
234 235	Pointer current screen color
236-238	Vector in keyboard decode
239	number of rharacters in keyboard buffer
240	Tyme of tape stlm
241	Bit 7=1 wRiit Bit 6=1 READ
275 277	Color luminrnam tamla in RAM
281 511	Prrogramer stack
51E 000	abmin input buffer
881-882	Previous abrir line number
883 884	Pointer Basir rirtrmode for CONT
754-755	Pointer tlmat to tlsed routine
756 757	Pointer tlab ir rntager
758-780	Vector basir acros Messages
770 771	Vector basir wart rtart

```
772-773        Vector addir token generator
774-775        Vector abrir list
776-777        Vector abrim  nonmaPe rremute
775-779        Vector abric token  nvmiate
788-781        Vector basir user token evaluirtr
788-789        Vector brakir keyadr
78A 78B        Vector prparr user token
78B 787        Vector interrupt
788 788        Vector hardware interrupt
78B 78L        Vector BRK interrupt
792-793        Vector formal RPRN
794 795        Vector  CLOSE
796-797        Vector CHKIN
795 79B        Vector CKOUNT
800 801        Vector  CLRCHN
802-803        Vector CHRIN
804-805        Vector CHROUT
8A8-8B7        Vector  STOP
8B8-8B9        Vector 8B1IN
810-811        Vector CLALL
812 813        Vector monitor break
814-815        Vector LOAD
818 817        Vector SAVE
818-1818       Irqa buffer
1138 114A      CHANGE1 subroutine
1145-1156      CHANGE subroutine
1883           Lart error number
1E2n 1A8       Row number of last error
1858-1867      88Scrence tpc  00 ESCAPE 0018
1A8            LEN jump mommePa
1881 1A8       LEN address tir/hi1
1A8            8A8 nmem value
1788 1788      Tabir rt inglial filr numberr
1788 1A8       Tabir of device numbers
1389 1310      Tabir of secondary addressmo
1310 138       Keyboard buffer
138n-138       Start address of RAM tor 88
1881-1838      Pointer  end of RAM 88
1338           Current color 0000
                 811   7 1-flash
                 811 6-4 luminrnam (8-71
                 811 3-0 color (8 181
1343           Size of keyboard buffer
134A           Flag Key repeat  $00=all  $t0=Nore  $88=SEL SPACE CURSORS
1348           Repeat t0000
1848           Repeat delay counter
1847           tlag  shitr rtri Cbm kay
1888           Lart pastor of rrirt
1848-1888      Pointer  keyboard tablr setup
1381           flag  88/71  $00=No  $A8=YES
1382           Prograna mounter 71
1383           Prarna mounter lr
1384           Processor flags
1885           Processor r reg
1886           Processor x reg
1387           Processor 7 reg
1888           Processor stack pointer
8838           Current Key pressed
```

LOW MEMORY MAP OF VM1

LABEL	dir	SECTIVAL	DESCRIPTION
$A818	$0000	0	8518 tirwriter registr
$5518	$0001	1	8818 I/8 Memory rPe trpe
A8881	$0007 0008	7 4	tlak tr tlak1 vector
$00882	$0000-0006	5-8	tlak0 to tlak1 vector
$0007	$0007	7	Search rharrtar
$A8C8R	$0008	8	End of quote tlag
$A7$$$	$0009	8	Save screen ltt1 1A8
$E88C	$000A	10	flag LOADing VERIFY=1
$8A8T	$000B	11	Pointer input buffer/rubscript
$1nFLB	$000C	12	$00=alt 81M to 10 flrt
$A11YP	$000D	13	addr type $7tring=FF Numeric=0
			Integer=1$8
$NTPLE	$000E	14	Float=0
$A8F8L	$000F	15	88TA mean/1151 quote/Garbage collert tlag
$LBFLE	$0010	10	Subscript/user t= call
$NFILE	$0011	17	Input tlag $00=input $40=Out $40=Read
$A8FLE	$0012	10	100 rign/pnnenrnon
	$0013	10	Current I/8 prompt
LINAM1	$A814-0815	20-21	Integer value
$8nPP1	$A818	82	Pointrr tems rtring rtack
LASTPT	$0017-0818	83-84	Last temy rtring address
TEMPPT	$0018 0001	88 33	Start for temy strings
$A8BX	$A8A8 0000	3n-37	Utility pointer rree
$E3A8	$A8A4-A8An	38-4B	Product area tor multiply
TFTPA8	$A8C1 A8CC	43-4n	Pointer rtrrt rf BASIC
$A8FTA8	$A8D8-A8EE	8A-4B	Pointer rirrt of variriles
$8FTA8	$A8F8	47-48	Pointer rirrt rt rrrage
$1F$A8	$A831-A83F	8A-4B	Pointer end of rrrrge
FRE7A8	$A833-A83n	51-52	Pointer ablom of rtrngs
$REA8	$A835-A836	53-54	Utility rirg pointer
$E7A81Z	$A837-A838	88-55	Pointer HigPoint rtArrea used by BASIC
CLRLIN	$A838 A83A	57-58	Current BASIC line number
$LSLIN	$A83B-A83C	88-8A	Previous BASIC line number
$LSTX1	$A83D A83E	8L-5B	BASIC strlement trr CONT
8LIN	$A83F-A848	8A-8A	Current BASIA line
$NTPT8	$A841 A843	8A-8L	Current DATA address
$NPT1B	$A8P3-A8P4	67-68	INPUT vector
$A8AA8T	$A8P5 A8P8	88-78	Pointer current variabir name
$A8PE1	$A8P7-A8P9	71 72	Pointer current variir data
$E8AP8	$A8A3-A8AA	73-74	Pointer variablr trr FOR/NEXI
	$A8AB $A8AC	75 76	Y-save/r-save/BASIC pointer save
	$A8AB	77	Comparisen $A8AA| accumulator
	$A8AE $A8B3	78-83	Mias emrk area
	$A8B4 A8B6	84-88	Jump vector tor tunctirne
	$A8B7 A8B8	87-88	Misr numerir emrk area
FACEXP	$A8B1	37	FPACC1 rxponent
FACSN	$A8B2-A8B3	88 181	FPACC1 - sentirss
FAC3$M	$A8B4	188	tPACC1 mign
$8PFLS	$A8B7	183	Pointer serlies evaluatinns constant
$LTS	$A8B8	1$1	FPACCR1 overflie digit
AREXP	$A8B5	188	FPACC2 exponent
ARSN	$A8BE-A8BF	188 188	FPACC2 sertisss
AREXSN	$A8BE	118	FPACC2 rise

```
$0050-$0061 80-B1       Usr pntr for FN defin. + for garb coll
$0052-$0054 B2-B4       Pntr accmplshr var list-string accesses
$0055       B6          Help flag $xx=HELP $xx=LIST
$0056-$0057 B7-B7       Jump vector for function evaluations
$0058       B8          Slnow
$0059       B9          Accss for LNSTH(B9 accr /temp pointer 1
$005A-$005B BA-BB       Pointer Blank transfer Str list
$005C-$0063 BC-C3       Pointer block transfer
$0065       C0          Temp pntr F  accumulatly floating-pt acc
                        8 pieces hsfmsx/afbc dac  for accumer
$0067       B7          Pntr acc  pt when reading Digit strings
$0068       B8          Exponent digit of the 8 acc: long FMBUI
$0069       B9          floating pt  accumulator 1 Mantissa
$006A-$006C B0-B2       floating pt  accumulator 1 Mantissa
$006D       B3          pointer  Polynomial evaluation
$006E       B4          floating pt  accumulator 2 Exponent
$006F       B5          floating pt  accumulator 2 Mantissa
$0070       B6          floating pt  acc  8 digit
$0071       B7          Result of flng digit compare acc 1 vs acc 2
$0072       B8          floating pt  accumulator 1 Round off
$0073-$0074 B9-C0       Pointer Cassette buffer
$0075-$0076 C1-C2       Offset value for AUTO increment 000=off
$0077       C3          Xtrns flag 1=8001C-start acc 1*8 higher
$0078       C4          Specify number counter for leading zeros
$0079       C5          Temp counter
$007A       C6          Temp fltr=age for indirect loading
$007B       C7          accumulation of accr-variable 000
$007C       C8          Tns of stack during program run
$007D       C9          Mode flag $xx=RUN mode $xx=direct mode
$007E       CA          USING pntr for 000 pnt  Stat 000 pointer
$007F       CB          Pntr2x
$0080       CC          Digits
$0081       CD          Current colour for graphic mode
$0082       CE          Multi-colour mode colour 1
$0083       CF          Multi colour mode colour 2
$0084       D0          Foreground colour
$0085       D1          X direction scale factor
$0086       D2          Y-direction scale factor
$0087       D3          Step drawing if not background colour
$0088-$0089 D4-D5       Address pointer for graphic routines
$008A       D6          Temp storage 1 for graphic routines
$008B       D7          Temp storage 2 for graphic routines
$008C       D8          Status word for kernal input/output
$008D       D9          Stny flag STOP key RVS key
$008E       DA          Time constant for cassette reads
$008F       DB          Load flag 000=LOAD 001=VERIFY
$0090       DC          Serial bus flag character in buffer
$0091       DD          Char in buffer for serial bus
$0092       DE          Sync # for save EOT maintained from tape
$0093       DF          Timeoutdev 4or0 address
$0094       E0          Index for file table no  of open files
$0095       E1          Standard input device 10 for keyboard=1
$0096       E2          Standard output device 13 for screen=3
$0097       E3          Parity byte from cassette
$0098       E4          Temp flng byte received
$0099       E5          Start flng for kernal
$009A       E6          Cassette error 000= 1 other error
$009B       E7          Cassette error type= 2 corrected
$009C-$009D E8-E9 168   Ptr for next free block in C/000 heap
$009E-$009F EA-EB 169   Temporary storage for serial bus
$00A0       EC          Countdown  SAVE on temp mar help mfr
$00A1       ED          Pointer for cassette buffer
$00A2       EE          Tape short counter RS-232 input block
$00A3       EF          Tape read err RS-232 receiver input bits
$00A4       F0          Tape B-read flag RS-232 start bit flag
$00A5       F1          Tape 0240 mode  00 for buffer input byte
$00A6       F2          Tape short counter 00 for input parity
$00A7-$00A8 F3-F4       Pointer Border accr: 1 cass buffer to/81
$00A9-$00AF F5-F6       pointer program end cassette pos Lo/01
$00AA-$00AB F7-F7       Cassette constant for flag
$00AC-$00B3 F8-F9       Pointer Start of cassette buffer Lo/01
$00B4       FA          Tpes bally pntr RS232 recv bit for accall
$00B5       FB          EOT order for RS-232 recv bit for transfer
$00B6       FC          Tpes help pointer 00 for byte buffer
$00B7       FD          Length of current Filenamed
$00B8       FE          Logical file number=LFN1
$00B9       FF          Current secondary address (SA)
$00BA       100         Current devices number (DA)
$00BB-$00BC 101-102     Pntr address of current filename Lo/81
$00BD       103         Temp pntr  RS-232 rotate parity buffer
$00BE       104         No  of remaining read/write Blanks
$00BF       105         Serial buffer
$00C0       106         Flag decsmitn second
$00C1       107         Start address in/Output (LO) Prank no
$00C2       108         Start address in/Output (HI) buffer no
$00C3-$00C4 109-10A     Tape LOAD temp pnt kernal vector address
$00C5       10B         Tape read/write data range
$00C6       10C         Temp no current LOAD SAVE VERIFY calls
$00C7       10D         Bank no  of current Filenamed 000 SRC
$00C8       10E         Pointer RS-232 input buffer
$00C9       10F         Pointer RS-232 output buffer
$00CA       110         Pointer keyboard decoder table
$00CB       111         Pntr to prtg pos  kernal PRINT routine
$00CC       112         Inmode for keyboard buffer accss
$00CD       113         Function key call flag
$00CE       114         Function key string call inmode
$00CF       115         @Diff Flag Shift=801 C000= Ctrl-001 old-000
$00D0       116         Flag for keyscreen
$00D1       117         Flag current processed key (C000)01=normal
$00D2       118         Flag for INPUT or GET   keyboard input
$00D3       119         Flag for 00/000 bottom mode
$00D4       11A         Flag for temP/quote/graphic mode
$00D5       11B         Pointer for char set RAM/ROM lonly bit 2)
$00D6       11C         Pointer for MSWL10 (lsi  <keyslr Bitmaps)
$00D7       11D         Pointer for MSWL10 (MSI  <keylsn answer>
$00D8       11E         Number of the function key
$00D9       11F         F key string length up to current F-key
$00DA       120         Temp F function key call<model>
$00DB       121         F key string length up to current (F key 1)
$00DC-$00DF 122-125     Pointer to running screen line text RAM
$00E0-$00E3 126-129     Pointer running screen line attributes RAM
$00E4       12A         Lower border of windnds
$00E5       12B         Upper border of windnds
$00E6       12C         Left border of windnds
$00E7       12D         Right border of windnds
$00E8       12E         Start of running input column
$00E9       12F         Offset of running input line
$00EA       130         End of running input line
$00EB       131         Current screen position line
$00EC       132         Current cursor position column
$00ED       133         Maximum number of screen lines
$00EE       134         Maximum number of screen columns
$00EF       135         Temp storage of characters to be put out
$00F0       136         Memory previous char for ESC temp
$00F1       137         Colour mode under screen for char output
$00F2       138         Colour mode protection for INSERT/DELETE
$00F3       139         Flag RVS mode active
$00F4       13A         Flag Quote mode active
$00F5       13B         Flag insert mode active
$00F6       13C         Flag Auto insert active
$00F7       13D         Cutoff switching of C-Shift 10001 obs
                        Ctrl 0 10=01
$00F8       13E         Cutoff of screens scrolling
$00F9       13F         Cutoff of 000 Forms mode by Ctrl 0
$00FA-$00FF 140-145     Free area for user applications
$0100       000         Inflow

$0002 - When using 64K C128 in CP/M mode the Zero page can be in
        the same be for the CPM
```

RS-232/VIC JUMP TABLE

LABEL	HEX ADD	DEC ADD	COMP	DESCRIPTIVE
	$FF79	65393	JMP $B7C2	Define function key no after HF0 addr after $22 F9 length in acc
	$FF7C	65360	JMP $BCA8	Print a word
	$FF7F	65359	JMP $F8B8	Print message
	$FF80	65360	JMP $F4A5	Call m C Monitor
	$FF83	65395		Wait mode
	$FF77	65400		obs mode
	$FF77	65407		obs mode
	$FF80	65408	$80	TOP mode
CINT	$FF81	65409	JMP $8884	Initialing editor
IOINIT	$FF82	65410	JMP $F3B0	Initialing I/O
RAMTAS	$FF87	65415	JMP $F35C	First RAM space cerd buffer rom ro SCOPE
RESTOR	$FF8A	65418	JMP $F8CE	Restore vectors
VECTOR	$FF8D	65421	JMP $F8E3	Vector RAM
SETMSG	$FF90	65424	JMP $F414	Control KERNAL messages
SECOND	$FF93	65427	JMP $F4F4	Send sec addr to listen
TKSA	$FF96	65430	JMP $EE10	Send sec addr to talk
MEMTOP	$FF99	65433	JMP $FA67	obs/read top RAM nplmfer
MEMBOT	$FF9C	65436	JMP $FA52	obs/read/set bottom RAM pointer
SCNKEY	$FF9F	65439	JMP $8811	Scan keyboard
SETTMO	$FFA2	65442	JMP $F4F8	Set timeout for (setting=1 IEC-Bus
ACPTR	$FFA5	65445	JMP $EE13	Input byte from serial port
CIOUT	$FFA8	65448	JMP $EDDF	Output byte via serial port
UNTLK	$FFAB	65451	JMP $ED09	Commabe serial bus to UNTALK
UNLSN	$FFAE	65454	JMP $EDFE	Commabe serial bus to UNLISTEN
LISTEN	$FFB1	65457	JMP $ED0C	Command ALL devices on bus to listen
TALK	$FFB4	65460	JMP $ED0F	Command devices on nprtrl bus to talk
READST	$FFB7	65463	JMP $FE1C	mode 1/0 status word
SETLFS	$FFBA	65466	JMP $FE00	Set logical filenr, who secondary addr
SETNAM	$FFBD	65469	JMP $FDF0	Determine file names
STEN	$FFC0	65472	JMP $E0100	Open a logical file	$EF93
CLOSE	$FFC3	64475	JMP $E031A4	Close a logical file	$E03E
CHKIN	$FFC6	65478	JMP $E0F1C	Open channel for input	$E010
CKOUT	$FFC9	65481	JMP $E0F64	Open channel for buspur	$E2B0
CLRCHN	$FFCC	65484	JMP $E0300	Close I/O channelr	$EF0C
CHRIN	$FFCF	65487	JMP $E00FF	Input character	$CE00
CHROUT	$FFD2	65490	JMP $E03D4	Output character	$EC40
LOAD	$FFD5	65493	JMP $F0F3	Load from peripheral device
SAVE	$FFD8	65496	JMP $F1CA	Store on peripheral device
SETTIM	$FFDB	65499	JMP $F0B8	Set time
RDTIM	$FFDE	65502	JMP $F0F8	Read time
STOP	$FFE1	65505	JMP $E0000	Scan STOP key	$F8E0
GETIN	$FFE4	65508	JMP $E0300	Peek char from KB buffer	$E3B6
CLALL	$FFE7	65511	JMP $E03A4	Close all channels and logical files
UDTIM	$FFEA	65514	JMP $CEF0	Increment Time
SCRORG	$FFED	65517	JMP $8863A	Identify X Y screen org ux
PLOT	$FFF0	65520	JMP $8880F	000/set X Y cursor positioning
IOBASE	$FFF3	65523	JMP $FC1A	Read addr-regnrs rom on 1 0 devices
	$FFF6	65526	JMP $FF3E	Switch on PSW
	$FFF9	65529	JMP $F8B4	Jump to reset routine
	$FFFC	65532	JMP $FFFF	Processor reset
	$FFFE	65534	JMP $FF48	Processor interrupt

USR KERNAL ROUTINES

HEX ADDRESS	DESCRIPTION
$E267	address of the operating system
$E2E8	Table for Commodore Key
$E2EC	Constants for CIA Timing
$E2EE	Sets 8-BIC address of the CIA or the VIA
$E2F5	Gets screen format line/column
$E2FA	Set number of get border position
$E310	Screen reset
$E32A	Clear screen
$E35B	Cursor mode
$E38B	Initialize VIC-II controller
$E3CF	Get character from keyboard buffer
$E3DA	Waiting loop for keyboard input
$E632	Get a character from the screen
$E684	Chanbe for quote
$E6B6	Calculate RAM for line start#
$E6DD	Irqlo of colour codes
$E700	Scroll screen
$E9C0	@Diff line up
$E9FF	Clear screen line
$EA1C	Set character and colour on screen
$EA4C	Calculate pointer on colour RAM
$EA31	Interrupt routine
$EA90	Keyboard prompt
$EB48	Checks on shift ctrl and commodore keys
$EBF5	Pointer on keyboard decoding tables
$EC01	Decoding tables
$EC44	Checks on control character
$EC78	Decoding tables
$EC88	Constants for video controller
$ED09	Send LISTEN
$ED0C	Send LISTEN
$ED40	Output of byte on IEC-Bus

```
$FFB0   Send secondary address for LISTEN
$FFB7   Send secondary address for TALK
$FFAE   Send UNTALK
$FFAB   Send UNLISTEN
$FF13   Set a byte free from IEEE bus
$EE13   Wait a millisecond delay
$FF96   Output RS232
$FF9A   Calculate number of RS232 data bits
$FF93   Output to RS232 buffer
$FFB6   GET of RS232
$FFA4   Set Timer for IEEE time-out
$FF90   Error messages of the operating system
af:90   Put out messages
$FF57   BASIC get a character
$FFCA   ASSIGN output a character
$FF9E   CHKIN Fixing of the input-device
$FF9B   CKOUT Fixing of the output-device
$FF81   CLOSE
$FF9F   Look for logical file number
$FF31   SET File parameter
$FF3F   CLALL closes all I/O channels
$FF2F   OPEN
$FF1F   LOAD
$FFA7   Output  Searching for file name
$FF8D   Output  Loading/verifying
$FF84   SAVE
$FF8A   Output  Saving filename
af:80   UDTIM increment running time
$FFDB   Set time
$FFDE   Set time
$FFE4   GET File Key
$FFD2   Put out error messages of the operating system
$FF7D   Read program header of tape
$FF76   Write header on tape
$FF74   Get start address of tape buffer
$FF77   Set start and end address of the tape buffer
$FF7F   look for name on tape header
$FF8D   increase tape buffer pointer
$FF17   Waits for tape key for reading
$FF38   Make for Tape key
$FF68   Waits for Tape Key for writing
$FF41   Send Files of tape
af:41   Read progress of Tape
$FF47   Write Tape buffer on TAPE
$FF49   Write Files of program on Tape
$FF80   Wait for I/O end
$FFE1   Checks on stop key
$FF6C   Read interrupt routine for mdem
$FF97   SET bit counter for serial output
$FFA5   Write on SIO on tape
$FFED   Write interrupt routine for Tape
$FF3D   SET IRQ vector
$FFC7   Switch off tape drive
$FFCC   Chanes on read/Write of end address
$FF3A   Increase Tenerne pointer
$FFE4   PEOT
$FF3D   Checks on DOS to SHOOT or SHOOS
$FF10   KEY-audio identify pattern
$FF19   Set or get hardware and I/O vectors
$FF84   Initialize Hardware and I/O vectors
$FF87   Initialize vorT memory
$FF84   Irbie of IRQ vectors
$FF8D   SET parameter for File names
$FF80   Set parameter for active File
$FF97   SET afafus
$FF1B   SET File for messages of the operating system
$FF1C   SET afafus
$FFE1   SET Timeout Flag for IEC end
af:E5   SET or get RAM-upper limit
$FF2A   SET or get RAM-lower limit
$FF13   NMI routine
$FFCE   Constants for RS232 baud rate
$FF10   Interrupt handler
```

JMP TABLE ADDRESSES

JMP ADDRESS	DESCRIPTION OF ROUTINE
$FF84	JMP $FD15 Initialize CIA's
$FF87	JMP $FD00 Clear or assign RAM
$FF8A	JMP $FD15 Initialize I/O
$FF8D	JMP $FF5B Initialize I/O vectors
$FF90	JMP $FF18 Set status
$FF93	JMP $FED2 Send SECOND secondary address
$FF96	JMP $FE97 Send TALK Secondary address
$FF99	JMP $FE98 Set/get TOP end
$FF9C	JMP $FE3A Get/get RAM start
$FF9F	JMP $FD30 Scan keyboard
$FFA2	JMP $FE01 Set IEC end Time out Time
$FFA5	JMP $FE13 Input for IEC end
$FFA8	JMP $FE67 Send UNTALK
$FFAE	JMP $ED0F Send UNLISTEN
$FFB1	JMP $ED36 Send LISTEN
$FFB4	JMP $EE8C Send LISTEN
$FFB7	JMP $EE97 Send TALK
$FFBA	JMP $FE00 Set File parameter
$FFBD	JMP $FE05 Set Filename parameter
$FFC0	JMP ($0314) OPEN $F34A
$FFC3	JMP ($0316) CLOSE $F291
$FFC6	JMP ($0318) CHKIN $F20E Set input device
$FFC9	JMP ($031A) CKOUT $F250 Set output device
$FFCC	JMP ($031C) CLRCH $F333
$FFCF	JMP ($031E) BASIN $F157 Input character
$FFD2	JMP ($0320) BSOUT $F1CA Output character
$FFD5	JMP ($0322) LOAD $F49E
$FFD8	JMP ($0324) SAVE $F5ED
$FFDB	JMP ($0326) Set time
$FFDE	JMP ($0328) Set Time
$FFE1	JMP ($032A) STOP $F6ED Scan stop-key
$FFE4	JMP ($032C) GETIN $F13E GET
$FFE7	JMP ($032E) CLALL $F32F
$FFEA	JMP ($0330) Increase Time
$FFED	JMP $F500 SCREEN get number lines and columns
$FFF0	JMP $F50A Set get cursor position
$FFF3	JMP $F500 Get start of I/O element
$FFFA	JMP $FE43 NMI vector
$FFFC	JMP $FCE2 RESET vector
$FFFF	JMP $FF48 IRQ vector

SOME KERNAL ROUTINES

HEX ADDRESS	DESCRIPTION OF ROUTINE
$F17C	Get attribute RAM address
$F0F0	Addresses of the individual writer code (Table)
$E5B5A	Base Table for Four number systems
$E3BE	Ball create tone
$F1F4	BSOUT output rest to screen
$E3B5	CLRN Send receive
$F81C	CrPAGE routine for F40 operations RAM
$F81C	CrPAGE routine for F4R operations ROM
$E259	Change IRQ vector for Tape operation
$E3F4	Check Commodore Key for Five delay
$F201	Check Filename for Cursor Band
$EA87	Clear From cursor position to screen end
$DA76	Clear From cursor position to line end
$E3BE	Clear From line start to line end
$E3B1	Clear line overline FIF
$E5BA	Commodore/Shift character set switch
$E5BE	Commodore/Shift switch to 40 column mdew
$E5BE	Commodore/Shift switch to 80 column Band
$E9C3	Copy RM1 one IBO routines to all Banks
$F7F3	CRSUI routine for 80 232 output
$F1EC	CRSUI evaluation on serial end
$F127	CRWIN evaluation on 80 232
$E7B6	CrWIN routine for F3 E99 input
$F1A8	CLOSE routine for Tape operation
$E0B0	Check cassette recorder Keyboard
$E3B0	Check Tape header address for veilBity
$E2E2	Check EXBON input Face cartridge test
$E619	Check 80 F80 Band parity
$EA8A	Clear or set auto-insert pointer
$E1AE	Clear screen window
$E5B0	Clear screen line in 40-column Band
$E9C8	Clear screen line in 80-column Band
$E8DE	Convert ans conVerte into end ASCII characters FX/F1
$E8E2	Convert ans in tab ASCII characters and output
$F7F6	Coprimate system FFFive word
$E2A8	Configure system as Commodore B4
$E5B0	Copy a window line (roll) to MBLINI
$E1B0	Copy a window line in 40 column mode
$E8B1	Body start address for Input/output operations
$F833	Control message output SFFACKING
$F54F	Control message output SFFRCKING FOR Filename
$F531	Control message output VERIFYING
$E5B5	Copy character to RAM into VIC RAM
$E3B8	Conversion From ASCII characters fr POKE mdew
$E3B0	Delete character under cursor
$E4B2	Delete current input line
$EA8A	Delete screen as window
$F1EA	Delete File entry From table
$F1C1	Adlere a File entry
$E31B	Adlete character to the left of the cursor
$E3BC	Adlete line on screen (with Shift move)
$E05B	Display monitor registers contents
$E0C5	Determine address of a monitor command
$E0A1	Determine address of BRANCH commands
$E1B0	Determine Tab position
$E3B0	Disable or enable Commodore/Shift
$E270	ENT ball routine of screen ADDR in RAM
$EAFE	Enable Fleft cursor
$F1B4	Editor IBO routine
$E5EF	Evaluate decoder table according to shift pattern
$E3B0	Evaluate ASC and stanf Keyscreen
$E7B6	Evaluate control mdew
$E1B0	Evaluate escape sequences
$E3E3	Evaluate insert
$E0B8	Fetch routine for F80 operations RAM
$F0B0	Fetch routine for F80 operations ROM
$F7C1	Fetch routine for F5V operations
$F7AE	Fetch routine for character From Filename
$E8E7	Flash VIC cursor
$E2E0	Function BAM tabl for C 180 mdew
$E5B0	Get Fit From decimal and into story Flag
$E0BA	Get cursor position and set
$E7A0	Get character From keyboard queue
$E5B0	Get character From colour of cursor position
$F2B0	Get character From screen
$F5EC	Get character From serial end
$EFA8	Get character From cassette
$E7B7	Get character Tape F0 F80
$F7CE	GET routine for F0 E80
$E3F3	GETIN evaluation for near keyboard
$E5B5	Give fast-mode pulse on serial end
$E2B0	Increment Tape buffer pointer
$E0B9	Initialize screen chs editor
$E0B8	Initialize editor and screen
$E0A5	Initialization of monitor commands
$E0B1	Initialize monitor For regular entry
$E01A	Initialize monitor after BREAK
$F1B0	Initialize VBC registers
$E37C	Insert line on screen
$FA69	Interrupt routine For Tape read
$E5B0	Interrupt routine for tape write
$EC2E	Insert FuncTion key string
$E0C5	JMPFAR routine RAM
$FB41	JMPFAR routine ROM
$A8C3	JSRFAR routine RAM
$FB5B	JSRFAR routine ROM
$E3A7	Jump to tab adge
$E4A0	Kernal Acafr routine
$EF8A	Kernal BASIN routine
$F334	kernal eadf routine
$FF7A	Kernal BSOUT routine
$F1B0	Kernal CHKIN routine
$F0B6	Kernal CHKIn routine
$E779	Kernal CHROUT routine
$E5B3	Kernal CIOut routine
$F1AC	Kernal CKOUT routine
$FCCC	Kernal CLALL routine
$F1B0	Kernal CLOSE routine
$FE66	Kernal CLRCH routine
$E795	Kernal BRA call routine
$E1AB	Kernal FSTMSBE routine
$F7EC	Kernal GETCPC routine
$EEE8	Kernal GetIN routine
$E2A0	Kernal GOBY routine
$F7B1	Kernal IRQMSE routine
$E1B0	Kernal IRINIT routine
```
```

```
ttttt7      Kernal IRQ routine
$C558       Kernal KEY routine (SHIFT to International versions)
$E3A3       Kernal LISTN routine
$F78F       Kernal LKUPLA routine
$F7B8       Kernal LKUPSA routine
$F286       Kernal LOAD routine
$F77L       Kernal MEMBOT routine
$F7B3       Kernal MEMTOP routine
$F12B       Kernal NMI routine
$AF1d       Kernal OPEN routine
$F86F       Kernal PHOENIX routine
tt017       Kernal QUIKSCN routine
$8053       Kernal RAMTBB routine
t7B0L       Kernal RDTIM routine
$F7A4       Kernal READST routine
tF730       Kernal RESET routine
$E500       Kernal SCNKB routine
tF737       Kernal SETBNK routine
$F7B0       Kernal SETLFS routine
$F75C       Kernal SETNAM routine
$F731       Kernal SETNAM routine
$F686       Kernal SETTIM routine
tF75L       Kegnal SETTMB routine
$E3B6       Kernel TALK routine
$F4B0       Kernel TIMB routine
$F510       Kernel UNTLK routine
$A8BE       Kernel UNLSN routine
0tS10       Kernal UNTLK routine
$C80C       Kernel USING routine
$F63E       Kernal VAUT routine
$F6BE       Kernel dIRP routine
$F8B8       Kernel dIRQ routine
$C82E       Key repeat evaluator
$C5E3       Keyboard matrix read
$F63D       Keyboard row selector  RUN/STOP RQ1?T
$C5E1       Keyboard scan evaluate
$C5CA       KeyPORTL buffer process for function key
$BF70       Load stack pointer and program counter time zero page
$1BF9       time energy from cassette
$F3Ea       LOAD switch in Direct mode
$F87?       time shuting from serial bus
$A6BE       Monitor command    temporaria a line)
$1B1B       Monitor command    (change register)
BB1eg       Monitor command > (change memory contents)
$BA00       Monitor command @ (disk command)
$BFd6       Monitor command : (compile a line)
$B13i       Monitor command C (compare memory areas)
$B639       Monitor command D  (disassemble memory)
$A31B       Monitor command F  (fill memory area)
$B100       Monitor command G  Jump to tXXt without return)
$B2CE       Monitor command H (HUNT for memory commands)
$B1BF       Monitor command J  Jump to tXXt with RTI)
$B307       Monitor command L  (load a program)
$B11E       Monitor command M  (display memory contents)
$1B5B       Monitor command R  (display register contents)
$B337       Monitor command T  (store t program)
$1B2A       Monitor command V  (compare program with memory)
$BE3       monitor command X  (quit)
$B0B1       Monitor command (Convert number to different system
tE805       NMI routine for RS-232
$EBF4       NMI routine for RS-232 output
$E27B       NMI routine for RS 232 input
$F818       Output baut sector message
$F9C0       Open file on serial bus
$F7F0       OPEN routine for tape operation
tF9+0       OPEN routine tan RS 232
$F75C       Output to RS tHE buffer
$CC0F       Output acc at screen position
tF819       Output contines accent
$CCE7       Output space on cursor position
$1SE2       Output byte on serial bus
$C76F       Output strolgen nature to screen
$C88C       Output characters at cursor position
$C7E9       Output underscter on screen
$F5E1       Output tround tilangeas or creaser
$F71E       Output byte and control messages
$CE9C       Passans byte output or serial bus
$F97F0      Prepare ace contents in two ASCII characters t 0Bt
$EAA1       Prepare cassette synchronization
$C303       Preture linefeed
$E0B0       Receive received bit from RS-232
$CCAE       Peogrom function key
$F4C6       Read data 0lora to burst mode
$F9FE       Read data 0lora tape tage
$F75A       Read data byte in burst mode
$CP58       Read an input line terminated by RETURN
$EE08       Read program header tape cassette
tt0B7       Recalculate tape bad addrdma
$BE57       Recandom operation bad
$EC80       Recorder motor aft
$C0e0       reset routine
$C651       Report Keyboard logic
$C770       Reset numba mode
$E000       reset CIAs to 00 tBE
$FE7e       Reset denumber cards Set portera
$1B83       Recasta BUS output buffer
$C0BE       reset tab stops
$1E5F       RS 232 request
$BB-t       RS-tBE data-bit number calculate
$1E72       RS tBE NMI status Set
$1E9A       RS tBE start bittest
$1tB7       RS 232 character output
$C30B       Scroll screen up
$F5C4       SAVE routine for tape operation
$1B5A       Search tape address for name
$C3C3       Search for bad of input line
$C1BF       Search in logical file number table
$CACA       Scroll down
tCAQC       Scroll up
$CA6C       Scrolling parmit or prohibit
ttC39       set attenuard I/O devices
$CA14       set allmode hardmen
$E3B5A      set bit counter for serial output
$CE37       Set or clear 0dd] pointer
$C3Fd       Set attributs addgress for attributs RAM
$E7L5       Sea character colour in 80-column mode
tt7EC       Sea character colour in 80-column mode
$C353       Set line averting bit
$C3B6       set nurser tlash mode
$C3S7       set cursor at current column
```

```
$C33E       Set nurser to end of line
$C1B8       Set nurser in screen almode at BASE position
$C3F5       Set nurser to left in almode to left
$C3B7       Set nurser up in almode
$C3F1       Set nurser right in almode
$C3B6       Set nurser down in almode
$CC88       Set nurser new position left in almode
$C3E3       Set nurser new position right in almode
$C33C       Set old nurser address again
$C3BE       Set cursor relief at nurser position
$F8B6       Set tilegrams to serial bus
$C3E1       Set or clear bas atop
$E373       Set clock frequency to tMHz
$C3BF       Sas or clear reverse mode
$C3E7       Set IRQ register
$F3B8       Sas program bas between after LOAD
$B2W        Stash routine for tBE operations RAM
$F3BE       Stash routine for tBE operations RAM
$F7IC       Stash routine for tBV operations
$C3BC       Switch 40/80 column opens
$FBA6       System IRQ routine
$FA+8       System NMI routine
$F7F8       Table of configuration values
$C1B8       Table at function key assignments
$C3BC       Table of function key codes
$EA6E       Table of IRQ patterns for tape operations
$CE74       Table of initialization values for 40 column
$C3BC       Table of initialization values for 80-column
$C73C       Table of serial codes
$8B+8       Table of PAL initialization values
$A4BC       Table of monitor keywords
$A4BE       Table of KIJPER assembler for RS-232 baud rate
$E8FB       Table for VDC initialization
$EDC7       Table for VIC initialization
$FCC3       TAPE memory Keys and reading commands
$C3F4       Test line averting bit
$C3BE       Test quote parameter and set pointer
$B7A6       Test separator between command operands
$EA8F       Test for STOP key
$A3B2       TSet for tape buffer
$C3BE       Turn off nurser flash mode
$C3I8       Turn nurser flash off for 40 column mode
$C3BE       Turn nurser flash on for 40 column mode
$C3BB       Turn nurser flash at tas 80 column mode
$C3F1       Turn nurser flash on for 80 column mode
$C3F8       Turn off 80 column reverse
$C3BE       Turn on 80 column reverse
$C3E3       Turn underline mode off
$C3C7       Turn underline mode on
$C3F1       Turn underline nurser on
$C3BE       Vector table to ASCII decodas tables
$FE3e       beator table to 010 decoder tables(alphabetical
            vertical only)
$A3BE       beator table for editor routines
$C3BE       Vector table for monitor routines
$E7B8       beator table for control mode routines
$1B6A       Verify routine in burst mode
$F47B       Wait for tape I/O termination
$F75C       Wait for bas of RS 232 transfer
$E6BC       Wait for fast mode assessme from bus
$E8CE       Wait for RECORD & PLAY on Cassette
$E9CA       Wait for button or Cassette
$E41 0      Write tape header to tape
$E3B8       Write bit to tape
$E918       Write data block to tape
$E91B       Write number to tape
$E41C       Write data block to tape
$E42F       Write the number
```

LOKPUL BASIC INTERPRETER ADDRESSES

HEX ADDRESS	DESCRIPTION OF ROUTINE
$A00E	Start vector
$A00E	NMI vector
$A00F	CBMBASIC
$A00E	Addresses of the BASIC command vtrue 1
$A3+E	Addresses of the BASIC functions
$A00E	Hierarchy-codes and HOOKcodes of the BASIC operations
$A00E	List of BASIC command units
$B11E	BASIC error messages
$B3B+	Messages of the BASIC interpreter
$A3BA	Stack cleanup routine for TBB NEXT and RUNUB
$B388	Block shifting routine
$A3F1	Chaoks on going in stack
$A00E	Opens space in memory
$A436	Output of out of memory
$A437	Output of error messages
$A00E	Break vector
$A47+	Handle parter
$A00E	Input routine loop
$A43C	Clear bas inserting program lines
$A533	Ile BASIC program lines asee
$A00E	Make a line into input buffer
$A671	Output of String tas long
$A57B	Chapas of a line into interpreter-ends
$A613	Look tor start addresas of a BASIC line
$A3+E	BASIC-command NEW
$A00E	BASIC-command CLR
$A00E	Set programs pointer to BASIC start
$A00E	BASIC-command LIST
$B717	Change interpreter mode to command per0
$B7A2	BASIC-command FOR
$B7AE	Interpreter loop, parsing out BASIC commands
$B706	Coering out BASIC command
$A81d	BASIC-command RESTORE
$0007	Interrupts progras at pressed stop-Key
$A82E	BASIC-command STOP
$A831	BASIC-command END
$A857	BASIC-command CONT
$A871	BASIC-command RUN
$A8E3	BASIC-command RUNUB
$A00E	BASIC-command BRIN
$A8F8	BASIC-command RETURN
$A8BC	BASIC-command DATA
$A00E	Looks for next statement
$A00E	Jumps for next line

```
$A068   BASIC-command LIST
$A09E   BASIC-command RUN
$A0E4   BASIC-command NEW
$A0F4   Looks for address of a BASIC line
$A13B   BASIC-command SET
$A154   BASIC-command CLR
$A15F   BASIC-command GOSUB
$A163   BASIC-command RETURN
$A171   BASIC-command DATA
$A177   Output string
$A19E   Output empty character (CR=carriage return)
$A1AB   Error handling for INPUT
$A1B8   BASIC-command NEXT
$A1DE   BASIC-command INPUT#
$A1E5   BASIC-command INPUT
$A206   BASIC-command READ
$A23C   Output 'Extra ignored' and 'Redo from start'
$A211   BASIC-command REDO
$A245   FRMNUM gets term and checks on numeric
$A248   Checks on numeric
$A24F   Checks on string
$A258   Output of 'Type mismatch'
$A262   FRMEVL gets and evaluates term
$A27E   Gets arithmetic term
$A286   floating point constant for PI
$A2B2   BASIC-command NOT
$A2F1   Gets term in parenthesis
$A2F7   Checks on Parenthesis closed
$A2FA   Checks on Parenthesis open
$A2FF   Checks on Comma
$A377   Checks on characters in accumulator
$A3A6   Output of 'Syntax error'
$A3B8   Gets variable
$A3E6   BASIC command GO
$A3F3   BASIC command GOTO
$A3FB   Compares two operands
$A400   BASIC command ON
$A413   Checks for 'Jump'
$A54A   Calculates pointer to first array-element
$A560   Floating point constant '32768'
$A57A   Change FAC to INTEGER
$A5B8   Output of 'Out of Data'
$A5CD   Output of 'Undef'd Function'
$A5E3   BASIC command DEF
$A5EC   Checks on FN syntax
$A5F4   BASIC-function FN
$A64B   BASIC function STR$
$A6C6   String space/allocation  Calculate pointer on string
$A6D7   Establishes string
$A6E5   Garbage collection  reserve unwanted strings
$A6DE   String correction
$A6DA3  String space allocation FRETMS
$A6EC   BASIC-function FRE
$A700   BASIC-function FRE
$A71E   BASIC function FRE
$A74C   BASIC function FIND78
$A787   BASIC-function HIGH
$A79C   BASIC-function LEFT
$A7D0   Get string parameter
$A7E0   BASIC function MID
$A7E0   Byte Mode term (0-255)
$A7E0   BASIC function VAL
$A828   Byte address (0 $3638) and byte value (2-255)
$A877   Change FAC to unknown format (Range 0-$3638)
$A886   BASIC function PEEK
$A8C4   BASIC command POKE
$A8D3   BASIC command WAIT
$A9D8   FAC = FAC - 0.5
$A9E0   Minus FAC = constant (F/V)   FAC
$A9E3   Minus FAC = MEM   FAC
$A9E7   Plus FAC = constant (A/Y)   FAC
$A9EA   Plus FAC = MEM = FAC
$A9F4   Output of 'Overflow'
$AA0C   Floating point constant for 0.5
$AA27   BASIC function LOG
$AA60   Multiplication FAC = constant (F/Y) = FAC
$AB28   Multiplication FAC = MEM = FAC
$AB5C   FAC = constant (F/Y)
$AB1E   FAC = FAC + 0.5
$AB4E   Floating point constant 10
$AB7E   FAC = FAC/10
$AB9F   FAC = constant (F/Y) / FAC
$ABA2   FAC = MEM/FAC
$ABB9   Output of 'Division by zero'
$ABBA   FAC = constant (A/Y)
$ABCA   Round FAC + FAC
$ABCC   Round FAC + .5
$ABD8   MEM = FAC
$ABDC   MEM = FAC
$ABE8   Round FAC
$BC2B   Get sign of FAC
$BC39   BASIC function SGN
$BC58   BASIC function ABS
$BC5B   Compare constant (A/Y) with FAC
$BC9B   Change from FAC to integer
$BCCC   BASIC function INT
$BCF3   Change ASCII to floating point
$BDDA   Floating point constants for floating point to ASCII
$BDCE   Output of line number at error messages
$BDC9   Output of positive integer Number (0-63999)
$BDD0   Change FAC to ASCII format
$BE11   Floating point constant 0.5
$BF1B   Binary numbers for change of FAC to ASCII
$BF71   BASIC function SQR
$BF78   FAC = constant (A/Y) to the power of FAC
$BF7B   FAC = SQR to the power of FAC
$BFAF   Floating point constant for EXP
$BFED   BASIC function EXP
$E043   Series 1 polynomial calculation
$E059   Series 2 polynomial calculation
$E08B   Floating point constant for RND
$E097   BASIC function RND
$E107   Output of 'Break'
$E16C   DEFAULT output of character
$E116   BASIC reads in a character
$E118   CHOUT establish output device
$E11B   CHKIN establish input-device
$E12A   GETIN get a character
$E13A   BASIC-command SYS
$E156   BASIC-command SAVE
$E165   BASIC-command VERIFY
$E1D0   BASIC-command LOAD
$E1DD   BASIC-command OPEN
$E1C7   BASIC-command CLOSE
$E1FA   Get parameters for LOAD and SAVE
$E219   Get par.Number for OPEN
$E2AD   BASIC-function CHR$
$E2B0   BASIC function STR$
$E2B4   BASIC function USR
$A464   BASIC function TAB
$E2DB   Floating point constants for SIN and COS
$E30D   BASIC-function ATN
$E33D   Floating point constants for ATN
$E37B   BASIC Ant jump-in
$E3A5   BASIC cold start
$E3AE   Copy of the CHRGET routine
$E3DA   Start value for the RND function
$E3EF   Initialize RAM for BASIC
$E4A7   Table of BASIC vectors
$E4ED   Load BASIC vectors
```

DISKETTE FORMAT & LAYOUT

BLOCK HISTRIBUTION BY TRACK

Track Numbers		Range of Settors		Total sectors		# Bloks	# Bl$AB
HEX	DEC	HEX	DEC	HEX	DEC		
$01	01-17	$00-$14	00-20	$15	21	YES	YES
$18 $00	18-24	$00-$11	00-10	$13	19	YES	YES
$19-$1E	25-30	$00-$10	00-17	$11	18	YES	YES
$1F-$23	31-35	$00-$0F	00-16	$11	17	YES	YES
$24 $2A	36-42	$00-$14	00-20	$15	21	$a	YES
$2B $30	43-48	$00-$13	00-19	$13	19	$a	YES
$31 $34	49-52	$00-$12	00-18	$13	19	$a	YES
$35 $41	53-65	$00-$11	00-17	$12	18	$a	YES
$42 $52	66-78	$00-$10	00-16	$11	17	$a	YES

SAM FORMAT (8x1 - TRACK 36 SECTOR 0

Byte number	Contents	Definition
0	18	Track of next Directory Block Always 18
1	1	Sector of next Directory Block Always 1
2	65	ASCII character 'A' indicating (2a1/81/71/4040 format
3		Double sided flag Ignored on 1541
4		Number of sector available on track 1
5		Track 1 sector 0-7 availability map
6		Track 1 sector 8-10 availability map
7		Track 1 Number 17-23 availability map
8		Number of Number available on track 2
9		Track 2 Number 0-7 availability map
10		Track 2 sector 8-10 availability map
11		Track 2 Number 17-23 availability map
ETC $04-$8F TB		
144		Number of sector available on track 36
141		Track 36 sector 0-7 availability map
142		Track 36 sector 8-10 availability map
143		Track 36 sector 17-23 availability map
$44-$80		Disk name (except with shifted spaces (CR=$a0)
$88-$81	$80	Shifted space (CR=$a0)
$82-$89		Disk ID
$8a $90	$80	Shifted space (CR=$a0)
$80 $80		ASCII representation of DA which are translatively the DAB version (2a format type) ($40/41/71/4040/2030
$87-$70		Shifted spaces (CR=$a0)
$71-$88		Nulls (CR=$0) not used

($71 Bytes as $00-$ unswept

0	Double sided DAB flag $00=Double $01=$0 $40=Single
$71-$88	Nulls (CR=$00) not used
$01-$37	Number of sector available track 36-52
	Needs easier by mode bytes!
$38 $	Number of sector available track 53
	(always 0 all sectors allocated)
$3A-$44	Number sector available if $4 $3Emode if by each bytes!
$45 $88	Number sector available tk $0-$5Eeach if by each bytes!
$81-$95	Number sector available tk $0-78(each if by each bytes!

PROGRAM FILE FORMAT

Byte	Definition
First SECTOR	
0 1	Track and number of next block in program file 1
2 3	Load address of program
4-255	Next 252 bytes of prg info stored as in comp mac (keywords tokenized)
SECOND-80 FULL SECTORS	
0 1	Track and number of next block in program file1
2-255	Next 254 bytes of prg info stored as in comp mac (Keywords tokenized)
Final SECTOR	
0 1	Null (0A0) followed by Number of valid data bytes in sector
2-???	Last bytes of prg info stored as in comp mac (keywords tokenized) The end of a BASIC file is marked by three zero bytes in a row. Any remaining bytes in last sector are garbage and may be ignored.

SEQUENTIAL FILE FORMAT

Byte	Definition
ALL BUT FINAL SECTOR	
0 1	Track and number of next sequential data block
2-255	254 bytes of data

```
SIGNAL VECTOR
% i          full (0000: followed by number of valid data bytes in sector
0-FFF        last bytes of data  Any remaining bytes are garbage & can be ignored
```

RELATIVE FILE FORMAT

BYTE	DEFINITION
DATA BLOCK	
0	Track and sector of next data block
1-255	254 bytes of data. Empty records contain 0FF (all binary ones) in the first byte followed by 000 (all binary zeroes) as the end of the record. Partially filled records are padded with nulls (00)
SIDE SECTOR BLOCK	
0-1	Track and sector of next side sector block
2	Side sector number (0-5)
3	Record length
4-5	Track and sector of first side sector (number 0)
6-7	Track and sector of second side sector (number 1)
8-9	Track and sector of third side sector (number 2)
10-11	Track and sector of fourth side sector (number 3)
12-13	Track and sector of fifth side sector (number 4)
14-15	Track and sector of sixth side sector (number 5)
16-255	Track and sector pointers to 120 data blocks

1541 FILE FORMAT,TRACK 18 SECTORS 1-18

BYTE	DIR ON 1130
0 1	track and sector of next directory block
2-4	file entry 1
5-83	file entry 2
84-98	file entry 3
99 127	file entry 4
128-130	file entry 5
131-161	file entry 6
162 193	file entry 7
194-256	file entry 8

STRUCTURE OF EACH INDIVIDUAL DIRECTORY ENTRY

BYTE	CONTENTS	DETAILED
0	128+type	File type 80 as with 000 as indicate properly closed file (if 00 as with 000 indicate file is locked) TYPES 0 = DELETED 1 = SEQUential 2 = PRGram 3 = USER 4 = RELative
1 2	Track and sector of first data block	
3-18	file name padded with shifted spaces	
19-20	Rel file only. Track and sector of first side sector	
21	Rel file only. Record length	
22 23	UNUSED	
24-27	Track and sector of replacement file during an @SAVE=@OPEN	
28 29	number of blocks in file. stored as a two-word integer in normal lo-byte hi byte format	

1541 MEMORY MAP

(Hex) <ADDRESS>

HEX	DECIMAL	DESCRIPTION
0000	0	Command used for buffer 0
0001	1	Command used for buffer 1
0002	2	Command used for buffer 2
0003	3	Command used for buffer 3
0004	4	Command used for buffer 4
0005 0007	5 7	Track and sector for buffer 0
0008 000D	8-0	track and sector for buffer 1
000E 0010	10-11	Track and sector for buffer 2
0011 0013	10-13	Track and sector for buffer 3
0014 0016	14-15	Track and sector for buffer 4
0018 001A	10-15	id for drive 0
001B 001C	00-01	ID for drive 1
001D-0017	02-03	ID
0020-0081	00-33	Flag for speed transport
0030-0031	48-48	Buffer pointer for disk controller
0032	57	Constant 0 part for begining of data block header
0033	08	Parity for data buffer
0034	01	Drive number for disk controller
0037	03	Buffer number for disk controller
0038	07	number of sectors per track for formatting
0039	71	Constant 7 part for begining of data block header
003A	73	Stack pointer
003B	71	step counter for speed transport
003C	01	Actual track number for formatting
003D 003E	105	disk area for sector division (id)
003F 0040	100	Number of read attempts (id)
0041-0070	111 118	Pointer to addresses for A and B commands
0077	118	Device number + 0000 128 used for listen
0078	1100	Device number + 0100 (400) for Talk
0079	121	Flag for listen (1/0)
007A	122	flag for talk (1/0)
007C	121	flag for 1/M from serial bus receiving
007D	100	flag for EOI from serial bus
007F	127	Drive number
0080	100	Track number
0081	129	Sector number
0082	130	Channel number
0083	131	Secondary address
0084	132	Secondary address
0085	133	Temp byte
0086 0088	138-141	Short storage for division
0089 008E	142-145	Actual buffer pointer
008F 0090	153-171	Address of buffer 0 00300
0091 009C	155 156	Address of buffer 1 00000
009D 009E	157 158	address of buffer 2 00500
009F 00A0	159 160	Address of buffer 3 00600
00A1 00A2	161-162	Address of buffer 4 00700
00A4-00A5	163 164	Pointer to input buffer 00200
00A6 00A8	100 168	Pointer to buffer error message 00C05
00A9-00AA	181 168	Record number LO block number LO
00AB 00C8	107 152	Record number HI block number HI
00C1 00C8	153 198	Write pointer for REL file
00C7 00CC	130-207	Record length for REL file
00D1	210	Pointer to record for REL file
00D2	213	Side sector number
00D3	214	Pointer to data block in side sector
00D7	215	Pointer to record in REL file
00D8	231	file type
00E7	240	Buffer number
00100-0147	256 325	Stack
00200-0228	012 000	Buffer for command string
00229	500	File type
00300	500	Record length
00238	001	Track side-sector
00239	002	Sector side-sector
00271	000	Length of input line
00278	030	Number of file opens
00257	063	file control method
00000-0000	000-004	Track of a file
00265 0000	005-009	Sector of a file
00000-0000	725-781	Buffer for error messages
00000-00FC	782 784	number at free blocks
00300-00FF	700-1023	Buffer 0
00400-01FF	1024 1279	Buffer 1
00500-00FF	1000 1400	Buffer 2
00000-00FF	1536 1791	Buffer 3
00700-07FC	1792 2047	Buffer 4

1541 DISK ERROR MESSAGES
000 THEIR CAUSES

The following list are the error messages that the DOS can recognize Note that TT and SS padded Track and Sector respectively

ERROR NUMBER	DESCRIPTION
00 00 00 00	This message occurs when the last disk operation was error free or it no command or data was sent after the last error occurred
00 READ ERROR TT SS	This error occurs that the header of a block was not found it is usually the result of a defective disk TT and SS designate the track and sector in which the error occurred Usually change the disk
01 HEAD ERROR T1 S5	The SYNC marker of a block was not found (the reason may be an unformatted disk or no disk in the drive This error can also be caused by misaligned read/write head Remedy Either insert a diskette and format the disk or move the header aligned
02 READ ERROR T1 S5	This error occurs that a checksum error has occurred in the header of a data block Just can be caused by the incorrect writing of a block
23 READ ERROR TT SS	This error implies that a data block was read into the DOS buffer but a checksum error occurs One or more data bytes are incorrect Remedy Have an ERROR This can sometimes be overcome by reading the data
24 READ ERROR TT SS	This error also results from a checksum error in the data block or in the preceding data header Incorrect bytes have been read Remedy Same as 23
25 WRITE ERROR TT SS	This is actually a VERIFY error After writing every block the data is read again checked against the data in the buffer This error is produced if the data are not identical Remedy Repeat the command that caused the error It this does not work find corresponding disk must be INITIALIZED out from further use by the Direct-Access Commands
26 WRITE PROTECT ON TT SS	An attempt was made to write to a disk with the write protect tab on Remedy Remove the tab
27 READ CHECK TT SS	A checksum error occurs in the header of a data block Remedy Repeat command or remove disk
28 WRITE ERROR TT SS	After writing a data block the SYNC characters of the next data block were not found Remedy Format the disk again or exchange it
29 DISK ID MISMATCH TT SS	The ID in the DOS memory does not agree with the ID on the diskette The diskette was either not initialized or there is an error in the header of a data block Remedy Initialize diskette
30 SYNTAX ERROR 00 00	A command was sent over the command channel that the DOS does not understand Remedy Correct her
31 SYNTAX ERROR 00 00	A command was not recognized by the DOS Remedy he not use the command
32 SYNTAX ERROR 00 00	The command sent over the command channel was over 40 characters Remedy Shorten the command
33 SYNTAX ERROR 00 00	A wildcard (* or ?) was used in an OPEN or SAVE command Remedy Remove wildcard
34 SYNTAX ERROR 00 00	The DOS cannot find the filename in a command The reason may be a colon was forgotten after the command word Remedy Check command
39 FILE NOT FOUND 00 00	User program of type USR was not found or automatic numbering Remedy Check filename
50 RECORD NOT PRESENT 00 00	A record was addressed in a relative data file that has not yet been written When writing a record this is not really an error You can avoid this error message if you write the highest record number of the file with OPEN(255) and initializing it this error will no longer occur upon later access
51 OVERFLOW IN RECORD 00 00	The number of characters sent when writing a record in a relative file was greater than the record length The excess characters are ignored
52 FILE TOO LARGE 00 00	The record number of a relative file is too big the diskette must have enough capacity Remedy buy another diskette or reduce the record number

00	WRITE FILE OPEN,00 00	An attempt was done to EMMA a file that had not been previously been CLOSED after writing. Usually the done 'f' in the DOS command re-done file
01	FILE NOT OPEN 00 00	a file was accessed that had not been OPENed Usually OPEN the file or check the filename
02	FILE NOT FOUND 00 00	An attempt was done to load a program or open a file that does not exist on the Disk Usually Check the filename
03	FILE EXISTS 00 00	An attempt was done to establish a new file with the name of an existing file on the Disk Usually Use @0:filename name or use 00
04	FILE TYPE MISMATCH 00 00	the file type used in the OPEN command does not agree with the file type in the Directory Usually Correct filetype
05	00 BLOCK TT 00	This message is given in association with the @:dsk allocate command when the amount+64 block is no longer free In this case the DOS automatically searched for a free block with a higher master and/or track number and given these values on the track and master number in the error message If no block with a greater number is free two zeroes will be given
06	ILLEGAL TT or 00 TT 00	It was attempt to use a block with the @:dsk commands that does not exist this error is then returned
07	ILLEGAL TT or 00 TT 00	The track-master designation TT a file produces a non-existent track or sector
70	00 CHANNEL 00 00	An attempt was made to open more files than channels available or a direct access channel is already reserved
71	010 FAT00 TT 00	The number TT in the 010 storage does not agree with the 64 Usually this means the disk has not been initialized
72	010 FULL 00 00	Fewer than three blocks are free on the disk or the various number TT Directory entries have been used 1144 on the 1041
73	CBM 000 v 00 1041 00 00	The message is the power up message of the 1041 As an error message it answers when an attempt is done to write to a disk that was not formatted with the same DOS version
74	0001 001 STUFF 00 00	The drive code not done a Disk:dir inserted
75	FORMAT SPEED ERROR 00 00	This error only occurs on the 000 ROM It indicates a deviation from the normal revolution speed whilst formatting

STANDARD 000 KEYWORD TOKENS

HEX	DEC		HEX	DEC		HEX	DEC	
000	00	SPACE	0+0	07	C	0B0	176	BIN
001	01	"	0+1	08	D	0B1	177	OCT
002	02	*	0+2	09	E	0B2	178	HEX$
003	03	#	0+3	10	F	0B3	179	POINTER
004	04	$	0+4	11	G	0B4	180	LEFT$
005	05	%	0+5	12	H	0B5	181	RIGHT$
006	06	&	0+6	13	I	0B6	182	MID$
007	07	'	0+7	14	J	0B7	183	GO
008	08	(	0+8	15	K	0B8	184	RESUME
009	09	)	0+9	16	L	0B9	185	TRAP
00A	10	*	0+A	17	M	0BA	186	TRACE
00B	11	+	0+B	18	N	0BB	187	SPRITE
00C	12	,	0+C	19	O	0BC	188	SPRITE$
00D	13	-	0+D	20	P	0BD	189	COLLISION
00E	14	.	0+E	21	Q	0BE	190	BUMP
00F	15	/	0+F	22	R	0BF	191	BUMP$
010	16	0	100	23	S	0C0	192	JOY
011	17	1	101	24	T	0C1	193	RSPPOS
012	18	2	102	25	U	0C2	194	RSPRITE
013	19	3	103	26	V	0C3	195	RSPCOLOR
014	20	4	104	27	W	0C4	196	XOR
015	21	5	105	28	X	0C5	197	DOT
016	22	6	106	29	Y	0C6	198	DOT$
017	23	7	107	30	Z	0C7	199	DEC
018	24	8	108	31	[	0C8	200	DEC$
019	25	9	109	32	\	0C9	201	HEX
01A	26	:	10A	33	]	0CA	202	HEX$
01B	27	;	10B	34	↑	0CB	203	COMBINE
01C	28	<	10C	35	←	0CC	204	ASC
01D	29	=	10D	36	!	0CD	205	HEX
01E	30	>	10E	37	"	0CE	206	HEX$
01F	31	?	10F	38	#	0CF	207	BIN
020	32	@	110	39	$	0D0	208	BIN$
021	33	A	111	40	%	0D1	209	IF
022	34	B	112	41	&	0D2	210	THEN
023	35	C	113	42	'	0D3	211	ELSE
024	36	D	114	43	(	0D4	212	GOTO
025	37	E	115	44	)	0D5	213	GOSUB
026	38	F	116	45	*	0D6	214	RETURN
027	39	G	117	46	+	0D7	215	REM
028	40	H	118	47	,	0D8	216	STOP
029	41	I	119	48	-	0D9	217	ON
02A	42	J	11A	49	.	0DA	218	WAIT
02B	43	K	11B	50	/	0DB	219	LOAD
02C	44	L	11C	51	0	0DC	220	SAVE
02D	45	M	11D	52	1	0DD	221	VERIFY
02E	46	N	11E	53	2	0DE	222	DEF
02F	47	O	11F	54	3	0DF	223	POKE
030	48	P	120	55	4	0E0	224	PRINT#
031	49	Q	121	56	5	0E1	225	PRINT
032	50	R	122	57	6	0E2	226	CONT
033	51	S	123	58	7	0E3	227	LIST
034	52	T	124	59	8	0E4	228	CLR
035	53	U	125	60	9	0E5	229	CMD
036	54	V	126	61	:	0E6	230	SYS
037	55	W	127	62	;	0E7	231	OPEN
038	56	X	128	63	<	0E8	232	CLOSE
039	57	Y	129	64	=	0E9	233	GET
03A	58	Z	12A	65	>	0EA	234	NEW
03B	59	[	12B	66	?	0EB	235	TAB(
03C	60	\	12C	67	@	0EC	236	TO
03D	61	]	12D	68	A	0ED	237	FN
03E	62	↑	12E	69	B	0EE	238	SPC(
03F	63	←	12F	70	C	0EF	239	THEN
040	64		130	71	D	0F0	240	NOT
041	65	A	131	72	E	0F1	241	STEP
042	66	B	132	73	F	0F2	242	+
043	67	C	133	74	G	0F3	243	-
044	68	D	134	75	H	0F4	244	*
045	69	E	135	76	I	0F5	245	/
046	70	F	136	77	J	0F6	246	↑
047	71	G	137	78	K	0F7	247	AND
048	72	H	138	79	L	0F8	248	OR
049	73	I	139	80	M	0F9	249	>
04A	74	J	13A	81	N	0FA	250	=
04B	75	K	13B	82	O	0FB	251	<
04C	76	L	13C	83	P	0FC	252	SGN
04D	77	M	13D	84	Q	0FD	253	INT
04E	78	N	13E	85	R	0FE	254	ABS
04F	79	O	13F	86	S	0FF	255	USR
050	80	P	140	87	T			
051	81	Q	141	88	U			
052	82	R	142	89	V			
053	83	S	143	90	W			
054	84	T	144	91	X			
055	85	U	145	92	Y			
056	86	V	146	93	Z			
057	87	W	147	94	[			
058	88	X	148	95	\			
059	89	Y	149	96	]			
05A	90	Z	14A	97	↑			
05B	91	[	14B	98	←			
05C	92	\	14C	99				
05D	93	]	14D	100	A			
05E	94	↑	14E	101	B			
05F	95	←	14F	102	C			
060	96		150	103	D			
061	97	A	151	104	E			
062	98	B	152	105	F			
063	99	C	153	106	G			
064	100	D	154	107	H			
065	101	E	155	108	I			
066	102	F	156	109	J			
067	103	G	157	110	K			
068	104	H	158	111	L			
069	105	I	159	112	M			
06A	106	J	15A	113	N			
06B	107	K	15B	114	O			
06C	108	L	15C	115	P			
06D	109	M	15D	116	Q			
06E	110	N	15E	117	R			
06F	111	O	15F	118	S			
070	112	P	160	119	T			
071	113	Q	161	120	U			
072	114	R	162	121	V			
073	115	S	163	122	W			
074	116	T	164	123	X			
075	117	U	165	124	Y			
076	118	V	166	125	Z			
077	119	W	167	126	[			
078	120	X	168	127	\			
079	121	Y	169	128	]			
07A	122	Z	16A	129	↑			
07B	123	[	16B	130	←			
07C	124	\	16C	131				
07D	125	]	16D	132	A			
07E	126	↑	16E	133	B			
07F	127	←	16F	134	C			

STANDARD CBM80 TOKENS

HEX	DEC		HEX	DEC		HEX	DEC	
080	128	END	0A0	160		0B5	181	ELSE
081	129	FOR	0A1	161		0B6	182	RESUME
082	130	NEXT	0A2	162	#	0B7	183	TRAP
083	131	DATA	0A3	163	/	0B8	184	TRON
084	132	INPUT#	0A4	164	↑ (POWER)	0B9	185	TROFF
085	133	INPUT	0A5	165	AND	0BA	186	SOUND
086	134	DIM	0A6	166	OR	0BB	187	VOL
087	135	READ	0A7	167	>	0BC	188	AUTO
088	136	LET	0A8	168	=	0BD	189	PUDEF
089	137	GOTO	0A9	169	<	0BE	190	GRAPHIC
08A	138	RUN	0AA	170	+	0BF	191	PAINT
08B	139	IF	0AB	171	-	0C0	192	CHAR
08C	140	RESTORE	0AC	172	*	0C1	193	BOX
08D	141	GOSUB	0AD	173	/	0C2	194	CIRCLE

128 DOUBLE BYTE TOKENS

PRE	SEC			HEX	DEC	
$CE	000	2	FAST	$0A	17	BLOCK
$CE	001	3	SLOW	$0B	18	RECORD
$CE	002	4	PEN	$0C	19	CONCAT
$CE	003	5	RSPPOS	$0D	20	DVERIFY
$CE	004	6	RSPRITE	$0E	21	DCLEAR
$CE	005	7	RSPCOLOR	$0F	22	SPRSAV
$CE	006	8	XOR	$10	23	CHKLIBRARY
$CE	007	9	DOT	$11	24	DELAY
$CE	008	10	DOT$	$12	25	BEGIN
$CE	009	11	DEC	$13	26	BEND
$CE	00A	12	DEC$	$14	27	WINDOW
$CE	00B	13	HEX	$15	28	BOOT
$CE	00C	14	HEX$	$16	29	WIDTH
$CE	00D	15	BIN	$17	30	SPRITE
$CE	00E	16	BIN$	$18	31	SPRSAV
$CE	00F	17	IF	$19	32	FETCH
$CE	010	18	THEN	$1A	33	PEN
$CE	011	19	ELSE	$1B	34	POT
$CE	012	20	GOTO	$1C	35	BAS
$CE	013	21	GOSUB	$1D	36	BUMP
$CE	014	22	RETURN	$1E	37	JOY
$CE	015	23	REM	$1F	38	RCLR
$CE	016	24	STOP	$20	39	RSPPOS
$CE	017	25	ON	$21	40	RSPRITE
$CE	018	26	WAIT	$22	41	RSPCOLOR
$CE	019	27	LOAD	$23	42	XOR
$CE	01A	28	SAVE	$24	43	DOT
$CE	01B	29	VERIFY	$25	44	DOT$
$CE	01C	30	DEF	$26	45	DEC
$CE	01D	31	POKE	$27	46	DEC$
$CE	01E	32	PRINT#	$28	47	HEX
$CE	01F	33	PRINT	$29	48	HEX$
$CE	020	34	CONT	$2A	49	BIN
$CE	021	35	LIST	$2B	50	BIN$
$CE	022	36	CLR	$2C	51	IF
$CE	023	37	CMD	$2D	52	THEN
$CE	024	38	SYS	$2E	53	ELSE
$CE	025	39	OPEN	$2F	54	GOTO
$CE	026	40	CLOSE	$30	55	GOSUB
$CE	027	41	GET	$31	56	RETURN
$CE	028	42	NEW	$32	57	REM
$CE	029	43	TAB(	$33	58	STOP
$CE	02A	44	TO	$34	59	ON
$CE	02B	45	FN	$35	60	WAIT
$CE	02C	46	SPC(	$36	61	LOAD
$CE	02D	47	THEN	$37	62	SAVE
$CE	02E	48	NOT	$38	63	VERIFY
$CE	02F	49	STEP	$39	64	DEF
$CE	030	50	+	$3A	65	POKE
$CE	031	51	-	$3B	66	PRINT#
$CE	032	52	*	$3C	67	PRINT
$CE	033	53	/	$3D	68	CONT
$CE	034	54	↑	$3E	69	LIST
$CE	035	55	AND	$3F	70	CLR
$CE	036	56	OR	$40	71	CMD
$CE	037	57	>	$41	72	SYS
$CE	038	58	=	$42	73	OPEN
$CE	039	59	<	$43	74	CLOSE
$CE	03A	60	SGN	$44	75	GET
$CE	03B	61	INT	$45	76	NEW
$CE	03C	62	ABS	$46	77	TAB(
$CE	03D	63	USR	$47	78	TO
$CE	03E	64	FRE	$48	79	FN
$CE	03F	65	POS	$49	80	SPC(
$CE	040	66	SQR	$4A	81	THEN
$CE	041	67	RND	$4B	82	NOT
$CE	042	68	LOG	$4C	83	STEP
$CE	043	69	EXP	$4D	84	+
$CE	044	70	COS	$4E	85	-
$CE	045	71	SIN	$4F	86	*
$CE	046	72	TAN	$50	87	/
$CE	047	73	ATN	$51	88	↑
$CE	048	74	PEEK	$52	89	AND
$CE	049	75	LEN	$53	90	OR
$CE	04A	76	STR$	$54	91	>
$CE	04B	77	VAL	$55	92	=
$CE	04C	78	ASC	$56	93	<
$CE	04D	79	CHR$	$57	94	SGN
$CE	04E	80	LEFT$	$58	95	INT
$CE	04F	81	RIGHT$	$59	96	ABS
$CE	050	82	MID$	$5A	97	USR
$CE	051	83	GO	$5B	98	FRE
$CE	052	84	MONITOR	$5C	99	POS
$CE	053	85	RENUMBER	$5D	100	SQR
$CE	054	86	SCNCLR	$5E	101	RND
$CE	055	87	DIRECTORY	$5F	102	LOG
$CE	056	88	DSAVE	$60	103	EXP
$CE	057	89	DLOAD	$61	104	COS
$CE	058	90	HEADER	$62	105	SIN
$CE	059	91	SCRATCH	$63	106	TAN
$CE	05A	92	COLLECT	$64	107	ATN
$CE	05B	93	COPY	$65	108	PEEK
$CE	05C	94	SWAP	$66	109	LEN
$CE	05D	95	DISPLAY	$67	110	STR$
$CE	05E	96	BANK	$68	111	VAL
$CE	05F	97	CONCAT	$69	112	ASC
$CE	060	98	DOPEN	$6A	113	CHR$
$CE	061	99	DCLOSE	$6B	114	LEFT$
$CE	062	100	RECORD	$6C	115	RIGHT$
$CE	063	101	HEADER	$6D	116	MID$
$CE	064	102	SCROLL	$6E	117	GO
$CE	065	103	SCALE	$6F	118	MONITOR
$CE	066	104	HELP	$70	119	RENUMBER
$CE	067	105	DO	$71	120	SCNCLR
$CE	068	106	LOOP	$72	121	DIRECTORY
$CE	069	107	EXIT	$73	122	DSAVE
$CE	06A	108	DIRECTORY	$74	123	DLOAD
$CE	06B	109	DSAVE	$75	124	HEADER
$CE	06C	110	DLOAD	$76	125	SCRATCH
$CE	06D	111	HEADER	$77	126	COLLECT
$CE	06E	112	SCRATCH	$78	127	COPY
$CE	06F	113	COPY	$79	128	SWAP
$CE	070	114	RENAME	$7A	129	DISPLAY
$CE	071	115	BACKUP	$7B	130	BANK
$CE	072	116	DELETE	$7C	131	CONCAT
$CE	073	117	RENAME	$7D	132	DOPEN
$CE	074	118	BACKUP	$7E	133	DCLOSE
$CE	075	119	DELETE	$7F	134	RECORD
$CE	076	120	RESET	$80	135	HEADER
$CE	077	121	MOVSPR	$81	136	SCROLL
$CE	078	122	SPRITE	$82	137	SCALE
$CE	079	123	SPRDEF	$83	138	HELP
$CE	07A	124	PUDEF	$84	139	DO
$CE	07B	125	CHAR	$85	140	LOOP
$CE	07C	126	BOX	$86	141	EXIT
$CE	07D	127	CIRCLE	$87	142	DIRECTORY
$CE	07E	128	PAINT	$88	143	DSAVE
$CE	07F	129	CHAR	$89	144	DLOAD
$CE	080	130	BOX	$8A	145	HEADER
$CE	081	131	CIRCLE	$8B	146	SCRATCH
$CE	082	132	PAINT	$8C	147	COLLECT
$CE	083	133	SSHAPE	$8D	148	COPY
$CE	084	134	GSHAPE	$8E	149	SWAP
$CE	085	135	WINDOW	$8F	150	DISPLAY
$CE	086	136	SCALE	$90	151	BANK
$CE	087	137	GRAPHIC	$91	152	CONCAT
$CE	088	138	SSHAPE	$92	153	DOPEN
$CE	089	139	GSHAPE	$93	154	DCLOSE
$CE	08A	140	WINDOW	$94	155	RECORD
$CE	08B	141	SCALE	$95	156	HEADER
$CE	08C	142	GRAPHIC	$96	157	SCROLL
$CE	08D	143	SSHAPE	$97	158	SCALE
$CE	08E	144	GSHAPE	$98	159	HELP
$CE	08F	145	WINDOW	$99	160	DO
$CE	090	146	SCALE	$9A	161	LOOP
$CE	091	147	GRAPHIC	$9B	162	EXIT
$CE	092	148	SSHAPE	$9C	163	DIRECTORY
$CE	093	149	GSHAPE	$9D	164	DSAVE
$CE	094	150	WINDOW	$9E	165	DLOAD
$CE	095	151	SCALE	$9F	166	HEADER
$CE	096	152	GRAPHIC	$A0	167	SCRATCH
$CE	097	153	SSHAPE	$A1	168	COLLECT
$CE	098	154	GSHAPE	$A2	169	COPY
$CE	099	155	WINDOW	$A3	170	SWAP
$CE	09A	156	SCALE	$A4	171	DISPLAY
$CE	09B	157	GRAPHIC	$A5	172	BANK
$CE	09C	158	SSHAPE	$A6	173	CONCAT
$CE	09D	159	GSHAPE	$A7	174	DOPEN
$CE	09E	160	WINDOW	$A8	175	DCLOSE
$CE	09F	161	SCALE	$A9	176	RECORD
$CE	0A0	162	GRAPHIC	$AA	177	HEADER
$CE	0A1	163	SSHAPE	$AB	178	SCROLL
$CE	0A2	164	GSHAPE	$AC	179	SCALE
$CE	0A3	165	WINDOW	$AD	180	HELP
$CE	0A4	166	SCALE	$AE	181	DO
$CE	0A5	167	GRAPHIC	$AF	182	LOOP
$CE	0A6	168	SSHAPE	$B0	183	EXIT
$CE	0A7	169	GSHAPE	$B1	184	DIRECTORY
$CE	0A8	170	WINDOW	$B2	185	DSAVE
$CE	0A9	171	SCALE	$B3	186	DLOAD
$CE	0AA	172	GRAPHIC	$B4	187	HEADER
$CE	0AB	173	SSHAPE	$B5	188	SCRATCH
$CE	0AC	174	GSHAPE	$B6	189	COLLECT
$CE	0AD	175	WINDOW	$B7	190	COPY
$CE	0AE	176	SCALE	$B8	191	SWAP
$CE	0AF	177	GRAPHIC	$B9	192	DISPLAY
$CE	0B0	178	SSHAPE	$BA	193	BANK
$CE	0B1	179	GSHAPE	$BB	194	CONCAT
$CE	0B2	180	WINDOW	$BC	195	DOPEN
$CE	0B3	181	SCALE	$BD	196	DCLOSE
$CE	0B4	182	GRAPHIC	$BE	197	RECORD
$CE	0B5	183	SSHAPE	$BF	198	HEADER
$CE	0B6	184	GSHAPE	$C0	199	SCROLL
$CE	0B7	185	WINDOW	$C1	200	SCALE
$CE	0B8	186	SCALE	$C2	201	HELP
$CE	0B9	187	GRAPHIC	$C3	202	DO
$CE	0BA	188	SSHAPE	$C4	203	LOOP
$CE	0BB	189	GSHAPE	$C5	204	EXIT
$CE	0BC	190	WINDOW	$C6	205	DIRECTORY
$CE	0BD	191	SCALE	$C7	206	DSAVE
$CE	0BE	192	GRAPHIC	$C8	207	DLOAD
$CE	0BF	193	SSHAPE	$C9	208	HEADER
$CE	0C0	194	GSHAPE	$CA	209	SCRATCH
$CE	0C1	195	WINDOW	$CB	210	COLLECT
$CE	0C2	196	SCALE	$CC	211	COPY
$CE	0C3	197	GRAPHIC	$CD	212	SWAP
$CE	0C4	198	SSHAPE	$CE	213	DISPLAY
$CE	0C5	199	GSHAPE	$CF	214	BANK
$CE	0C6	200	WINDOW	$D0	215	CONCAT
$CE	0C7	201	SCALE	$D1	216	DOPEN
$CE	0C8	202	GRAPHIC	$D2	217	DCLOSE
$CE	0C9	203	SSHAPE	$D3	218	RECORD
$CE	0CA	204	GSHAPE	$D4	219	HEADER
$CE	0CB	205	WINDOW	$D5	220	SCROLL
$CE	0CC	206	SCALE	$D6	221	SCALE
$CE	0CD	207	GRAPHIC	$D7	222	HELP
$CE	0CE	208	SSHAPE	$D8	223	DO
$CE	0CF	209	GSHAPE	$D9	224	LOOP
$CE	0D0	210	WINDOW	$DA	225	EXIT
$CE	0D1	211	SCALE	$DB	226	DIRECTORY
$CE	0D2	212	GRAPHIC	$DC	227	DSAVE
$CE	0D3	213	SSHAPE	$DD	228	DLOAD
$CE	0D4	214	GSHAPE	$DE	229	HEADER
$CE	0D5	215	WINDOW	$DF	230	SCRATCH
$CE	0D6	216	SCALE	$E0	231	COLLECT
$CE	0D7	217	GRAPHIC	$E1	232	COPY
$CE	0D8	218	SSHAPE	$E2	233	SWAP
$CE	0D9	219	GSHAPE	$E3	234	DISPLAY
$CE	0DA	220	WINDOW	$E4	235	BANK
$CE	0DB	221	SCALE	$E5	236	CONCAT
$CE	0DC	222	GRAPHIC	$E6	237	DOPEN
$CE	0DD	223	SSHAPE	$E7	238	DCLOSE
$CE	0DE	224	GSHAPE	$E8	239	RECORD
$CE	0DF	225	WINDOW	$E9	240	HEADER
$CE	0E0	226	SCALE	$EA	241	SCROLL
$CE	0E1	227	GRAPHIC	$EB	242	SCALE
$CE	0E2	228	SSHAPE	$EC	243	HELP
$CE	0E3	229	GSHAPE	$ED	244	DO
$CE	0E4	230	WINDOW	$EE	245	LOOP
$CE	0E5	231	SCALE	$EF	246	EXIT
$CE	0E6	232	GRAPHIC	$F0	247	DIRECTORY
$CE	0E7	233	SSHAPE	$F1	248	DSAVE
$CE	0E8	234	GSHAPE	$F2	249	DLOAD
$CE	0E9	235	WINDOW	$F3	250	HEADER
$CE	0EA	236	SCALE	$F4	251	SCRATCH
$CE	0EB	237	GRAPHIC	$F5	252	COLLECT
$CE	0EC	238	SSHAPE	$F6	253	COPY
$CE	0ED	239	GSHAPE	$F7	254	SWAP
$CE	0EE	240	WINDOW	$F8	255	DISPLAY
$CE	0EF	241	SCALE			
$CE	0F0	242	GRAPHIC			
$CE	0F1	243	SSHAPE			
$CE	0F2	244	GSHAPE			
$CE	0F3	245	WINDOW			
$CE	0F4	246	SCALE			
$CE	0F5	247	GRAPHIC			
$CE	0F6	248	SSHAPE			
$CE	0F7	249	GSHAPE			
$CE	0F8	250	WINDOW			
$CE	0F9	251	SCALE			
$CE	0FA	252	GRAPHIC			
$CE	0FB	253	SSHAPE			
$CE	0FC	254	GSHAPE			
$CE	0FD	255	WINDOW			
$CE	0FE	256	SCALE			
$CE	0FF	257	GRAPHIC			

FILER ABBREVIATED KEYWORDS

KEYWORDS	ABBREVIATION	KEYWORD	ABBREVIATION	KEYWORD	ABBREVIATION
000	Ashift@	000.0	@Ashift@	READ	Rshift@
ABC	Ashift@	0010	25+1@	RENAME	REshift@
ATN	Ashift@	GRAPHIC	Gshift@	RENUMBER	RE%shift@
AUTO	Ashift@	GSHAPE	25+1@	RENUMBER	RE%shift@
BACKUP	25+1@	HEADER	H25+1@	RETURN	RE%shift@
BANK	25+1@	HEX$	H@shift@	RETURN	RE%shift@
CHAR	Cshift@	INPUT#	Ishift@	RETURN	RE%shift@
CHR$	Cshift@	INPUT	Ishift@	RETURN	RE%shift@
CMD	Cshift@	LOAD	Lshift@	RIGHT$	Rshift@
CIRCLE	Cshift@	LEFT$	Lshift

ABOVE	Shift£5	INSTR	Ishift$	RUN	Shift$
ABS	Shift%	JOY	Jshift$	RANDOM	Shift$
ATN/ND	Cshift$	KEY	Kshift$	SAVE	Sshift$
CHAR	CShift$	LEFT$	tshift$	SCALE	SShift$
CHR$	Cshift$	LEN	Lshift$	SCRCLR	Sshift$
CIRCLE	Cshift$	LIST	Lshift$	SCRATCH	SShift$
CLOSE	CLshift$	LOAD	Lshift$	SGN	Sshift$
CLR	Cshift$	LOCATE	Lshift$	SID	Sshift$
CMD	Cshift$	LOOP	Lshift$	SLOOP	Sshift$
COLLECT	COLLshift$	WIDE	Pshift$	SOUND	Sshift$
COLLISION	CShift%	MONITOR	MShift$	SPECLOAD	SPshift$
COLSE	CLshift$	MOVSPR	Mshift$	SPPEEK	SPshift$
CONCAT	Cshift$	NEXT	Nshift$	SPRITE	Sshift$
COPY	Cshift$	NO	Nshift$ ON	SPRSAV	SPshift$
DATA	Dshift$	ON	Oshift$ OR	SQR	Sshift$
DCLEAR	DCLshift$	OPEN	Oshift$	SQUARE	Sshift$
DCLOSE	Dshift$	PAINT	Pshift$	STASH	Sshift$
DELETE	Dshift$	PEEK	Pshift$	STEP	Sshift$
DIR	Dshift$	PEN	Pshift$	STRP	Sshift$
DIRECTORY	Dshift$	PLAY	Pshift$	STR$	Sshift$
DLOAD	Dshift$	POKE	Pshift$	SWAP	Sshift$
DOPEN	Dshift$	POT	Pshift$	TAB(	Tshift$
DSAVE	DAshift$	PRINT	?	THEN	Tshift$
DVERIFY	Dshift$	PRINT#	Pshift$	TRAP	Tshift$
DVERIFY	Dshift$	PRINT USING	?USshift$	TROFF	TRshift$
ENVELOPE	tshift$	PUDEF	Pshift$	TRON	TRshift$
ERR$	Xshift$	RGRAPH	RGshift$	UNTIL	Ushift$
EXIT	tshift$	RCLR	Rshift$	USR	USshift$
EXP	Eshift$	RDOT	Rshift$	USRIFY	Ushift$
FETCH	tshift$	READ	Rshift$	VOL	Vshift$
FILTER	tshift$	RECORD	Rshift$	WHIT	Wshift$
FIX	tshift$	RENAME	Rshift$	WAIT	Wshift$
				WIDTH	Wshift$
				WINDOW	WSshift$
				XOR	Xshift$

NOTE: Any keywords not listed indicates that there is not abbreviation for that particular keyword

FLASH COLOUR CODES

Value to POKE for each colour

COLOUR	VALUE
Black	1
White	0
Red	3
Cyan	4
Purple	5
Green	6
Blue	7
Yellow	8
Orange	9
Brown	10
Yellow-green	11
Pink	12
Blue-green	13
Light blue	14
Dark blue	15
Light green	16

80% SCR/COLOUR CODES & POKES

Value to POKE for each colour

COLOUR	LOW NYBBLE FOR COLOUR VALUE	HIGH NYBBLE FOR COLOUR VALUE	MULTICOLOUR COLOUR VALUE
Black	0	0	0
White	1	10	0
Red	2	20	10
Cyan	3	30	11
Purple	4	40	12
Green	5	50	13
Blue	6	60	14
Yellow	7	70	15
Orange	8	80	
Brown	9	90	
Lt Red	10	A0	
Dark Grey	11	B0	
Med Grey	12	C0	
Lt Green	13	D0	
Lt Blue	14	E0	
Lt Grey	15	F0	

Where to POKE colour values for each mode

MODE	BIT No BIT PAIR	LOCATION	COLOUR VALUE
Pattern test	0	53281	Low nybble
	1	Colour memory	Low nybble
Multicolour	00	53281	Low nybble
text	01	53282	Low nybble
	10	53283	Low nybble
	11	Colour memory	Select multicolour
Extended	00	53281	Low nybble
colour text	01	53282	Low nybble
!!!	10	53283	Low nybble
	11	Colour memory	Low nybble
Bitmapped	0	Screen memory	Low nybble !21
	1	Screen memory	High nybble !21
Multicolour	00	53281	Low nybble !23
bitmapped	01	Screen memory	High nybble !21
	10	Screen memory	Low nybble !23
	11	Colour memory	Low nybble

!03 For all modes the screen border colour is controlled by POKE 53280 with the low nybble colour value

!!! In extended colour mode bits 6 & 7 of each byte in screen memory serve as the bit pair controlling background colour. Because only bits 0-5 are available for character selection only characters with screen codes 0 to 63 can be used in this mode

!2) In the bitmapped modes the high and low nybble colour values are read together and POKed into the SAME LOCATION in screen memory to control the colours of the corresponding CELL in the 8 x 8 map. For example to control the colours of cell 0 of the bitmap 00 the high and low nybble values are POKE the result into location 0 of screen memory

CIA6 COLOUR CODES

COLOUR SOURCE

NUMBER	SOURCE
0	40 column background colour (CIE)
1	Foreground for graphics border (CIE)
2	Foreground colour 1 for multicolour screen (CIC)
3	Foreground colour 0 for multicolour screen (CIC)
4	38 column border (CIC whether in text or graphics mode)
5	Character colour for 40 or 80 column text screen
6	80 column background colour (D54A)

Value to POKE for each colour 40 column mode

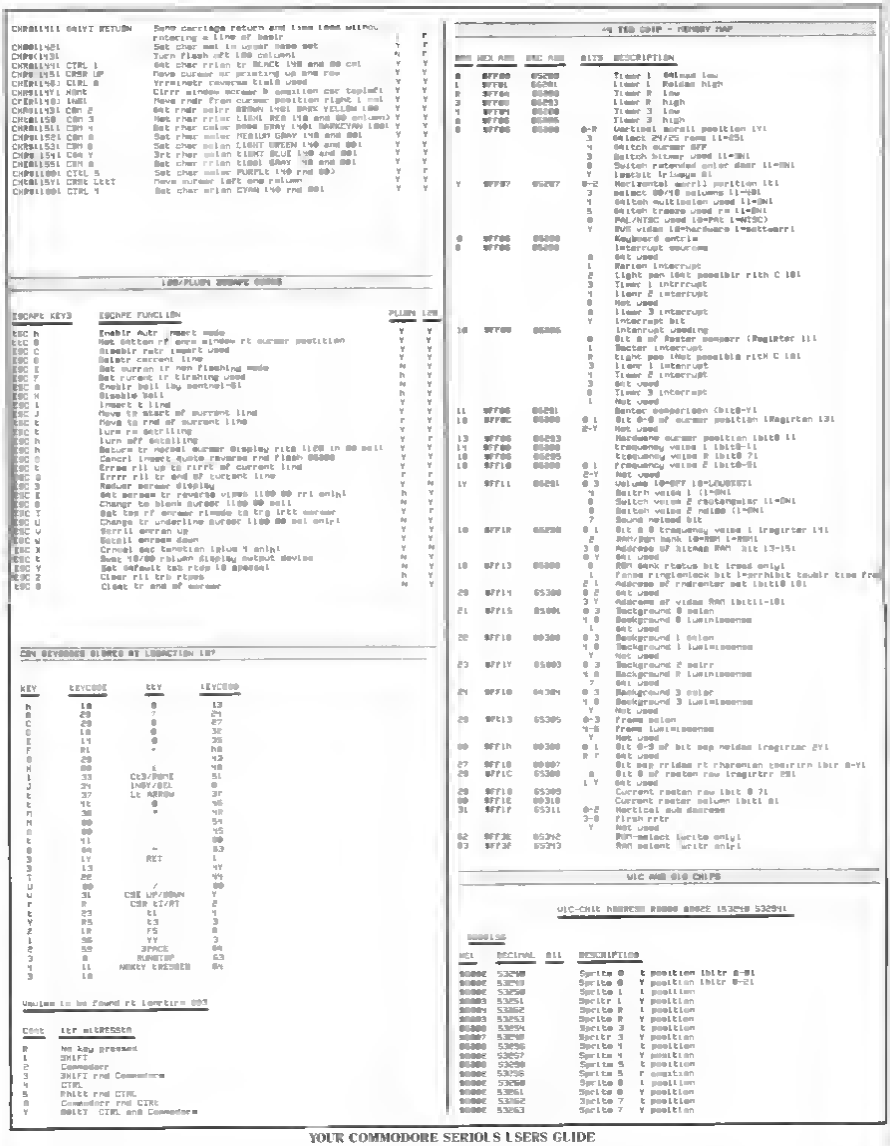
COLOUR	VALUE
Black	1
White	0
Red	5
Cyan	4
Purple	0
Green	0
Blue	7
Yellow	8
Orange	0
Brown	10
Light red	11
Dark grey	10
Medium grey	15
Light green	14
Light blue	10
Light grey	10

Value to POKE for each colour 80 column mode

COLOUR	VALUE
Black	1
White	2
Dark red	3
Light cyan	4
Light purple	5
Dark green	6
Dark blue	7
Light yellow	8
Dark purple	9
Brown	10
Light red	11
Dark cyan	12
Medium grey	13
Light green	14
Light blue	15
Light grey	16

CBM CONTROL & CHR$ CODES

CHR$	KEY!!	FUNCTION	C64	C128
CHR$(2)	CTRL B	Underline ON !80 col	N	Y
CHR$(5)	CTRL 2/CTRL E	Set char color to white 80 col 40 col	Y	Y
CHR$(7)	CTRL G	Produce bell tone	N	Y
CHR$(8)	CTRL H	Disable char set change key	Y	N
CHR$(9)	CTRL I	Enable char set change key	Y	N
		Move cursor to next tab position	N	Y
CHR$(10)	CTRL J	Sets cursor to next line	N	Y
		Send a line feed	N	Y
CHR$(11)	CTRL K	Disable char set change key	N	Y
CHR$(12)	CTRL L	Enable char set change key	N	Y
CHR$(14)	CTRL N	Send set #1 (lg load # enter) ENGLISH	Y	Y
CHR$(15)	CTRL O	ONE char set to lower/upper case set	Y	Y
CHR$(17)	CTRL Q	Turn cursor on 80 col	N	Y
CHR$(18)	CTRL R/CBM9	Move cursor down one row	Y	Y
CHR$(19)	CTRL S	Chars to be printed in reverse field	Y	Y
CHR$(20)	CTRL 0/HOME	Move cursor to home position	Y	Y
CHR$(20)	CTRL 1/DEL	Delete last char typed	Y	Y
CHR$(21)	CTRL 7/ESC	Same as ESC character	N	Y
CHR$(28)	CTRL 3/CTRL /	Make char color RED !40 and 80 column	Y	Y
CHR$(29)	CTRL 5/CBM8	Move cursor one column to right	Y	Y
CHR$(30)	CTRL 9	Set char color to GREEN !40 and 80 column	Y	Y
CHR$(34)	"	Print double quote on scn & set quote mode	Y	Y
CHR$(129)	CBM 1	Set char color ORANGE !40 col UNIX PURPLE 80 col		
CHR$(130)		Underline off 80 column	N	Y
CHR$(131)		Run a programme (only works from kb buffer)	Y	Y
CHR$(132)	tl	Reserved CHR$ code for tl key	Y	Y
CHR$(133)	F3	Reserved CHR$ code for t3 key	Y	Y
CHR$(134)	F5	Reserved CHR$ code for F5 key	Y	Y
CHR$(135)	t1	Reserved CHR$ code for F7 key	Y	Y
CHR$(137)	t2	Reserved CHR$ code for t2 key	Y	Y
CHR$(138)	F4	Reserved CHR$ code for F4 key	Y	Y
CHR$(139)	F6	Reserved CHR$ code for F6 key	Y	Y
CHR$(140)	F8	Reserved CHR$ code for F8 key	Y	Y



CHR$(141) CR$(Y RETURN Same carriage return and line feed as when entering a line of basic
CHR$(142) Set char set to upper case set
CHR$(143) Turn flash off (CBM onlyset)
CHR$(144) CTRL 1 Set char color brian tr BLACK (+0 and 00 on)
CHR$(145) CRSR UP Move cursor or printing up one row
CHR$(146) CTRL 0 Yrrminate reverse field used
CHR$(147) CLR Clrrr window screen P position cor topleft
CHR$(148) Ins Move rndr from cursor position right 1 col
CHR$(149) CBM E Set char colrr brown (+01 DARK YELLOW (+00
CHR$(150) CBM 3 Set char prior light red (+0 and 00 onlyset)
CHR$(151) CBM 4 Set char color DARK GRAY (+01 MARKCYAN (+01
CHR$(152) CBM 8 Set char color MEDIUM GRAY (+0 and 001
CHR$(153) CBM 9 Set char colon light GREEN (+0 and 001
CHR$(154) CBM Y Set char colon light BLUE (+0 and 001
CHR$(155) CBM @ Set char rrian light GRAY (+0 and 001
CHR$(180) CTRL 5 Set char color PURPLE (+0 rnd 00)
CHR$(157) CRSR LEFT Move cursor left one column
CHR$(180) CTRL 7 Set char brian CYAN (+0 rnd 001


```
D018  53272             9th bit of sprite X co-ordinate
                    0   Sprite 0
                    1   Sprite 1
                    2   Sprite 2 etc through sprite 7
D011  53265         VIC Control Register
                    7   Raster compare register  Bit 8
                    6   1=Enable extended color text mode
                    5   1=Enable bit map mode
                    4   1=Blank screen to border
                    3   1=25 row text display  0=24 row text display
                  F 0   Smooth scroll to Y dot position
D012  53266         Raster compare register  Position of raster on screen
D013  53267         Light pen X position
D014  53268         Light pen Y position
D015  53269         Enable or disable sprite
                    0   1=Enable sprite 0
                    1   1=Enable sprite 1
                    2   1=Enable sprite F etc through sprite 7
D016  53270         VIC Control Register
                    7   Unused
                    6   Unused
                    5   1=Full colour mode on
                    4   1=Multicolour mode on
                    3   1=40 Column text  0=38 Column text
                    2-0 Smooth scroll to X position
D017  53271         Double Vertical Expansion
                    0   Double sprite 0 Vertically
                    1   Expand sprite 1 Vertically
                    F   Double sprite F Vertically etc through to sprite 7
D018  53272         VIC Memory Control
                    7 4 Video matrix mode address
                    3-0 Character set mode address
D019  53273         VIC Interrupt Flags
                    7   set to any VIC IRQ condition
                    3   light pen latch mode (bit 7)
                    2   Sprite sprite triggered (bit 7)
                    1   Sprite vs background triggered (bit 7)
                    0   Raster compare triggered (bit 7)
D01A  53274         VIC Interrupt Enables
                    3   1=Enable light pen interrupt
                    2   1=Sprite vs sprite enabled
                    1   1=Sprite vs background enabled
                    0   1=Raster compare enabled
D01B  53275         Sprite Priority Registers
                    0   1=Sprite 0 passes in front of graphics
                    1   1=Sprite 1 passed in front of graphics etc through to 7
D01C  53276         Sprite multi-colour select
                    0   1=Sprite 0 is multicolour
                    1   1=Sprite 1 is multicolour etc through sprite 7
D01D  53277         Sprite Horizontal Expansion
                    0   1=Sprite 0 expanded horizontally
                    1   1=Sprite 1 expanded horizontally etc through sprite 7
D01E  53278         Sprite vs sprite collision detection  if a sprite is
                    touching a sprite  the bit for that sprite is turned on
                    Sprite vs background  if sprite has hi text or
                    background  relevant bit is set
                    Border colour
                    Background colour
D021  53281         Multi colour 1
D022  53282         Multi colour 2
D023  53283         Multi colour 3
D024  53284         Sprite multi-colour
D025  53285         Double multi colour
D026  53286         Sprite 0 colour
D027  53287         Sprite 1 colour
D028  53288         Sprite 2 colour
D029  53289         Sprite 3 colour
D02A  53290         Sprite 4 colour
D02B  53291         Sprite 5 colour
D02C  53292         Sprite 6 colour
D02D  53293         Sprite 7 colour
D02E  53294         C128 ONLY REG2
                  0 48=sprites in the C128 contains as 2Mhz or 1Mhz
                    if the bit is set then there is no access in the VIC
                    chip and the C128 operates in the 2 Mhz mode
```

SID CHIP ADDRESS D400-D41C (54272-54300)

ADDRESS			
HEX	DECIMAL	BIT	DESCRIPTION
D400	54272		Voice 1 low byte of frequency
D401	54273		Voice 1 high byte of frequency
D402	54274		Voice 1 low byte of pulse width
D403	54275	3-0	Voice 1 high byte of pulse width
D404	54276		Voice 1 Control Register
		7	1=Sawtooth voice
		6	1=Pulse waveform on
		5	1=Sawtooth waveform on
		4	1=Triangle waveform on
		3	1=Disable voice 1
		2	1=Ring modulate voice 1 with voice 3
		1	1=Synchronize voice 1 with from of voice 3
		0	1=Start attack decay sustain 0=Start release
D405	54277		Voice 1 Attack/decay
		7-4	Attack cycle duration
		3-0	decay cycle duration
D406	54278		Voice 1 Sustain/release
		7-4	Sustain cycle duration
		3-0	release cycle duration
D407	54279		Voice 2 low byte of frequency
D408	54280		Voice 2 high byte of frequency
D409	54281		Voice 2 low byte of pulse width
D40A	54282	3-0	Voice 2 high byte of pulse width
D40B	54283		Voice 2 Control Register
		7	1=Random noise on
		6	1=Pulse waveform on
		5	1=Sawtooth waveform on
		4	1=Triangle waveform on
		3	1=Disable oscillator 3
		2	1=Ring modulate oscillator 2 with oscillator 1
		1	1=Synchronize oscillator 2 with oscillator 1 frequency
		0	1=Start attack decay sustain 0=Start release
D40C	54284		Voice 2 Attack/decay
		7-4	Attack cycle duration
		3-0	decay cycle duration
D40D	54285		Voice 2 Sustain/release
		7-4	Sustain cycle duration
		3-0	Release cycle duration
D40E	54286		Voice 3 low byte of frequency
D40F	54287		Voice 3 high byte of frequency
D410	54288		Voice 3 low byte of pulse width
D411	54289	3-0	Voice 3 high byte of pulse width
D412	54290		Voice 3 Control Register
		7	1=Random noise on
		6	1=Pulse waveform on
		5	1=Sawtooth waveform on
		4	1=Triangle waveform on
		3	1=Disable voice
		2	1=Ring modulate oscillator 3 with from of oscillator 2
		1	1=Synchronize oscillator 3 with from of oscillator 1
		0	1=Start attack decay sustain 0=start release
D413	54290		Voice 3 Attack/decay
		7-4	Attack cycle duration
		3-0	decay cycle duration
D414	54292		Voice 3 Sustain/release
		7-4	Sustain cycle duration
		3-0	release cycle duration
D415	54293	2-0	Filter cut off low middle
D416	54294		Filter cut-off high byte
D417	54295		Filter Control
		7-4	Filter resonance
		3	1=External input to Filter
		2	1=Voice 3 to Filter
		1	1=Voice 2 to Filter
		0	1=Voice 1 to Filter
D418	54296		Filter Volume And Mode
		7	1=Turn off voice 3 output
		6	1=High pass Filter on
		5	1=Band pass Filter on
		4	1=Low pass Filter on
		3-0	Output volume
D419	54297		A/D converter for POT X
D41A	54298		A/D converter for paddle 2
D41B	54299		Produces random number when voice 3 set to noise
D41C	54300		Output of voice 3 envelope generator

6510 PROCESSING MODES & OPCODES

The following table gives the HEX values for the various OPcodes in their individual addressing modes The following key to OP used for the Addrmode mod

```
#    = Accumulator
#    = Immediate
ZP   = Zero mode
RR   = Relative
ABS  = Absolute a
ABY  = Absolute Y
ZP1  = Zero mode 1
ZPY  = Zero mode Y
( X) = Indexed X
( Y  = Indexed Y
```

MNEMONIC	ADDRESSING MODE								
--	#	(	ZP	RR	ABS	ABY	ZP1	ZPY	,X)	) Y
ADC		RR	69	60	78	79	75	-	61	71
AND	-	RR	69	60	2R	69	35	-	E1	31
ASL	0A	-	06	00	1E	-	16	-	-	-
BIT	-	-	24	AC	-	-	-	-	-	-
CMP	-	CD	C5	C8	8R	89	89	-	C1	81
CP1	-	88	E4	AC	-	-	-	-	-	-
CPY	-	CB	C4	AC	-	-	-	-	-	-
DEC	-	-	CA	CE	80	-	80	-	-	-
DNE	-	48	45	48	80	80	55	-	41	51
INC	-	-	E6	60	78	-	FE	-	-	-
LDK	-	RR	A9	A0	8R	80	80	-	A1	81
LDK	-	A8	A6	A8	-	80	-	80	-	-
LDY	-	A0	A4	AC	-	-	B4	-	-	-
LSK	4A	-	46	4E	8R	-	80	-	-	-
ONE	-	09	05	RR	10	10	10	-	01	11
PHL	2A	-	RR	PE	3E	-	36	-	-	-
NOK	9A	-	RR	80	7E	-	78	-	-	-
SBC	-	E0	F8	88	F8	F8	F8	-	E1	F1
STA	-	-	80	80	80	80	80	-	81	81
STK	-	-	86	80	-	-	-	80	-	-
STX	-	-	B4	AC	-	-	B4	-	-	-

6502 INSTRUCTIONS

BRANCH INSTRUCTIONS

BPL	BMI	BVC	BVS	BCC	BCS	BNE	BEQ
10	30	RR	78	RR	RR	D0	F0

TRANSFER INSTRUCTIONS

TXA	TAX	TYA	TAY	TSX	TXS
RR	A0	98	RR	RR	9A

STACK INSTRUCTIONS

PHP	PLP	PHA	PLA
08	28	RR	RR

JUMP INSTRUCTIONS

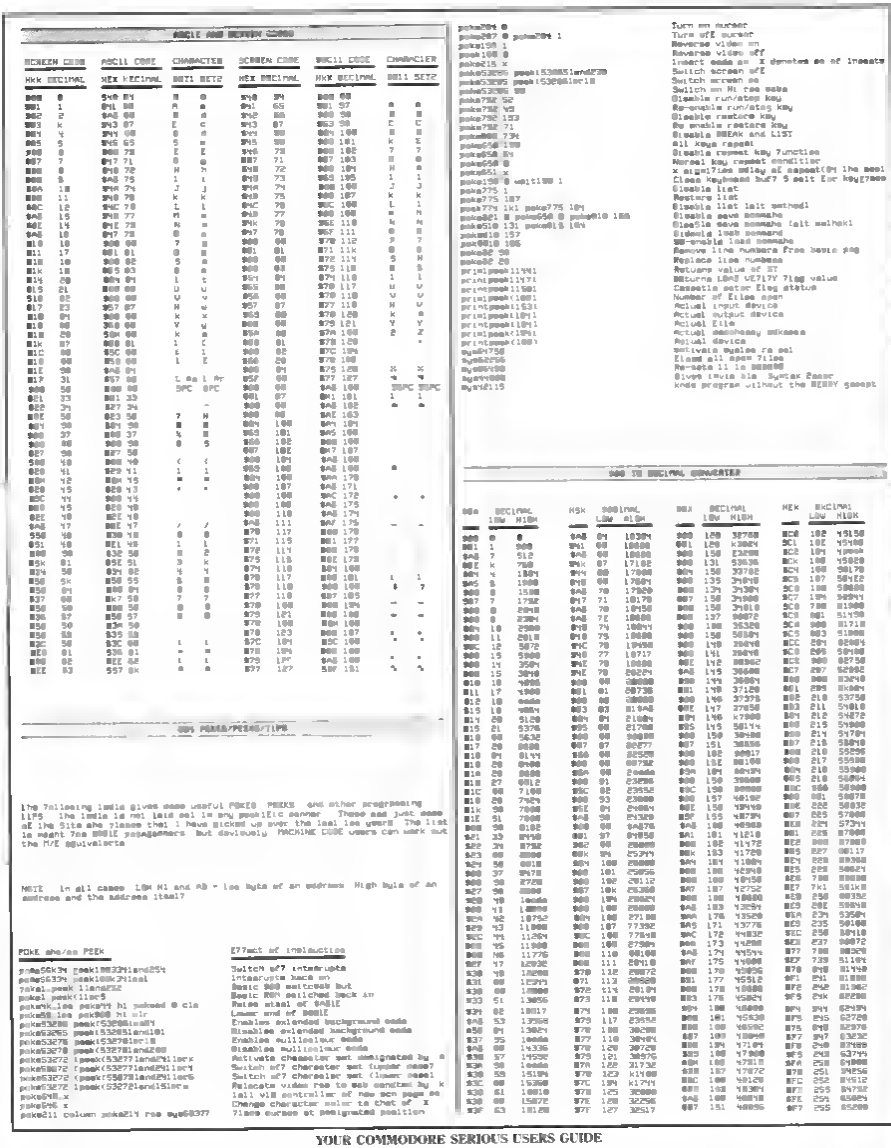
BRK	JSR	RTI	RTS	JMP	JMP	BRK
RR	80	RR	RR	4C	6C	EA

FLAG INSTRUCTIONS

CLC	SEC	CLI	SEI	CLV	CLD	SED
18	38	RR	78	B8	RR	F8

INC/DEC INSTRUCTIONS

DEY	INY	DEX	INX
88	C8	CA	RR



ASCII AND SCREEN CODES

SCREEN CODE	ASCII CODE	CHARACTER	SCREEN CODE	ASCII CODE	CHARACTER
Hex Decimal	Hex Decimal	Bit1 Bit2	Hex Decimal	Hex Decimal	Bit1 Bit2
00 0	40 64	@	20 32	60 96	
01 1	41 65	A	21 33	61 97	a
02 2	42 66	B	22 34	62 98	b
03 3	43 67	C	23 35	63 99	c
04 4	44 68	D	24 36	64 100	d
05 5	45 69	E	25 37	65 101	e
06 6	46 70	F	26 38	66 102	f
07 7	47 71	G	27 39	67 103	g
08 8	48 72	H	28 40	68 104	h
09 9	49 73	I	29 41	69 105	i
0A 10	4A 74	J	2A 42	6A 106	j
0B 11	4B 75	K	2B 43	6B 107	k
0C 12	4C 76	L	2C 44	6C 108	l
0D 13	4D 77	M	2D 45	6D 109	m
0E 14	4E 78	N	2E 46	6E 110	n
0F 15	4F 79	O	2F 47	6F 111	o
10 16	50 80	P	30 48	70 112	p
11 17	51 81	Q	31 49	71 113	q
12 18	52 82	R	32 50	72 114	r
13 19	53 83	S	33 51	73 115	s
14 20	54 84	T	34 52	74 116	t
15 21	55 85	U	35 53	75 117	u
16 22	56 86	V	36 54	76 118	v
17 23	57 87	W	37 55	77 119	w
18 24	58 88	X	38 56	78 120	x
19 25	59 89	Y	39 57	79 121	y
1A 26	5A 90	Z	3A 58	7A 122	z
1B 27	5B 91	[	3B 59	7B 123	+
1C 28	5C 92	£	3C 60	7C 124	
1D 29	5D 93	]	3D 61	7D 125	-
1E 30	5E 94	↑	3E 62	7E 126	π
1F 31	5F 95	←	3F 63	7F 127	◥
20 32	60 96	L as L Rv	40 64	80 128	SHIFT SHIFT
21 33	61 97		41 65	81 129	
22 34	62 98		42 66	82 130	
23 35	63 99		43 67	83 131	
24 36	64 100		44 68	84 132	
25 37	65 101		45 69	85 133	
26 38	66 102		46 70	86 134	
27 39	67 103		47 71	87 135	
28 40	68 104		48 72	88 136	
29 41	69 105		49 73	89 137	
2A 42	6A 106		4A 74	8A 138	
2B 43	6B 107		4B 75	8B 139	
2C 44	6C 108		4C 76	8C 140	
2D 45	6D 109		4D 77	8D 141	
2E 46	6E 110		4E 78	8E 142	
2F 47	6F 111		4F 79	8F 143	
30 48	70 112		50 80	90 144	
31 49	71 113		51 81	91 145	
32 50	72 114		52 82	92 146	
33 51	73 115		53 83	93 147	
34 52	74 116		54 84	94 148	
35 53	75 117		55 85	95 149	
36 54	76 118		56 86	96 150	
37 55	77 119		57 87	97 151	
38 56	78 120		58 88	98 152	
39 57	79 121		59 89	99 153	
3A 58	7A 122		5A 90	9A 154	
3B 59	7B 123		5B 91	9B 155	
3C 60	7C 124		5C 92	9C 156	
3D 61	7D 125		5D 93	9D 157	
3E 62	7E 126		5E 94	9E 158	
3F 63	7F 127		5F 95	9F 159	

```
poke204 0                    Turn on cursor
poke207 0 poke204 1          Turn off cursor
poke198 1                    Reverse video on
poke198 0                    Reverse video off
poke215 x                    Insert mode on  X denotes no of inserts
poke53265 peek(53265)xor$F20 Switch screen off
poke53265 peek(53265)or$10   Switch screen on
poke53280 99                 Switch on RL rev mode
poke772 52                   Disable run/stop key
poke772 53                   Re-enable run/stop key
poke772 71                   Disable restore key
poke772 71                   Re enable restore key
poke808 234                  Disable SHIFT and LIST
poke808 237                  All keys repeat
poke650 0                    Disable repeat key functions
poke650 64                   Repeat key, cursor controller
x argm[1]=s smlay af appeat(@x) the mesl
Clear Keyboard buf? 5 salt Esc key77=no
Disable list
Restore list
Disable list jalt method1
Disable save nonsmple
Disable save nonsmple jalt method1
Disable load nonsmple
UN-enable load nonsmple
Remove line numbers free basic prg
Replace line numbers
Returns value of ST
Returns LOAD VERIFY flag value
Causes motor flag status
Number of files open
Actual input device
Actual output device
Actual File
Actual dev/temp, mknsss
Actual device
activate duplex re sel
Clear all open files
Re-call it is nnnnnn
Gives twice the  Syntax Error
kres program without the READY prompt
```

```
poke53272,(peek(53272)and251)or4
poke53272,(peek(53272)and251)or12
poke53272,(peek(53272)and251)or8
poke53272,(peek(53272)and251)or0
poke[ff],x  poke[ff],y  poke[ff],z
poke[ff],x  poke[ff],y  poke[ff],z
poke[ff],x  poke[ff],y  poke[ff],z
```

Foreign Formats

In theory, the C128 has the valuable ability to read a wide range of CP/M disk formats. In practice, it isn't so easy. Your Commodore shows you how to do it.

One of the most interesting features of the C 128's CP/M mode is its ability to read a wide number of MFM disk formats created on other CP/M systems. The feature was only provided so that the 128 owner would have ready access to the full range of CP/M applications without having to rely on companies to copy programs on to the non standard Commodore format diskette. By contrast owners of, for example, Amstrad machines are in the main limited to a selection of the most popular CP/M packages.

While this in itself is a major benefit, provision of the facility also provides some less well-publicised advantages. Of course it means that you can now create your own programs and datafiles for use on other CP/M machines. What however will be of interest to a greater number of 128 owners is that the disk drives work faster with MFM disks than with the standard GCR format. The main reason for this is almost certainly, because the physical track/sector layout of Comodore disks does not fit very well with CP/M's internal logical representation of a disk. Additionally the new disk ROM routines for handling MFM disks are the only tide sections within the disk ROMs.

Horror movie

At the point it would be very easy to digress into a discussion of CP/M internals and the horror movie lurking within your disk drive. For readers interested in these topics I have suggested books on both topics at the end of this article and we will just consider the more immediate problem of creating an MFM format disk.

Further investigations have revealed that while Commodore originally intended at one time to provide an MFM formatting facility from within CP/M it is now extremely difficult to do so. Fortunately however it is relatively simple to do this using BASIC 7.0.

In order to implement CP/M, the new disk drives support a set of instructions called Burst Commands. These account for the slight improvement in disk performance that is always obtained in CP/M mode by using faster data transfers. The main functions in this group are for fast read and write of 128 byte CP/M logical records and a special fast program LOAD command. Also included within the set is a general purpose MFM formatting command. The information on this is found on page 84 of the 1571 Disk Drive Users Guide. It isn't too hard to use Burst Commands — I successfully formatted a disk to KAYPRO II format at the second attempt.

Straightforward

Everything seemed quite straightforward so I was somewhat surprised when every other format I attempted to create was greeted by CP/M with the response MISSING. This is shorthand for "I have searched through my internal tables, oh master, and cannot find an entry that matches this format". Clearly another gem from the Commodore School of Technical Authorship. The disk manual seems to be accurate, but gives no details on the actual formats supported. They are given in the CP/M sections of the main manual, but this does not tell you how the sectors are numbered which you need to identify.

There is one place where you can find this information and you will only have it if you have bought the CP/M utilities and documentation pack. This package also contains a disk of CP/M sources and at the end of the BIOS file CXDISK.ASM you will find Disk Parameter Block (DPB) Table. The DPB Table holds preset information for a range of different file formats, but is modifiable by the user.

After the DPB entries for the Commodore formats and the MFM formats there are a number of blank entries and two unimplemented ones used on Morrow machines. This means it is possible to insert new entries without having to discard any of the existing ones. The easiest way to modify this file is to use a word processor otherwise you will have to struggle with the infamous ED editor.

The DPB table is unique to Commodore, a most CP/M machines have only a single DPB. This is required by the operating system in order to convert from the logical structure of the disk used in the BDOS section and the physical structure of tracks and sectors on the disk. Logically CP/M considers a disk to contain a number of 128 byte records, which are organised into blocks of between 1k and 16k bytes in length. For the C128 the block size is 1k for single-sided disks and 2k for double sided disks.

By Paul Schofield

This is an important parameter as it defines the minimum size of a CP/M disk file regardless of how little data it may contain

The parameters in the DPB are used to calculate the numbers of the sectors on a particular track that correspond to a logical block For formatting purposes the only parameters of interest are the sector size the number of sectors per track and the number allocated to the first sector on a track The other parameters concern the order in which the disk drive writes sectors on the disk. These parameters can be specified in the format command, but they should not be required

The demonstration program shown in Listing 1 will create two of the most useful formats The KAYPRO II format packs more data than normal on a disk and the IBM 8 formats are very widely used Using the table of parameters for other formats provided in Table 1, it would of course be possible to extend the program to allow the creation of all the supported formats However, unless you are planning to provide a CP/M disk copying service, it is probably better to limit the formats you are using

MFM Formatting Parameters Table

No.	Format	Sides	Sct size (B5)	Sct/Trk (B7)	1st Sct (B3)
1	Epson QX10	2	1	16	129
2	Epson QX	2	2	10	129
3	IBM 8 SS	1	2	8	129
4	IBM 8 DS	2	2	8	129
5	KAYPRO IV	2	2	10	128
6	KAYPRO II	1	2	10	128
7	Osborne DD	2	3	5	129
8	Osborne SD	1	3	5	129
9	Epson Euro	2	1	16	129

Formats 10-16 are available for user definition

Further reading

As mentioned earlier here are a couple of book recommendations for readers who would like to pursue these subjects further A very good introduction to this subject is provided by *CP/M The Software Bus* This is a programmer's companion written by Andrew Clarke, Mike Eaton and David Powys Lybbe and published by Sigma Technical Press This book scores high on readability and follows a sensible progression from the very basics through editors, assemblers and compilers to the operating system internals

I have only two reservations in recommending this to C128 users The first is that the book concentrates on CP/M rather than CP/M Plus which is used on the 128. While the differences are fully covered but not in great detail Secondly in my copy at least (several years old) the chapter on CP/M programming languages badly needs updating and does not even mention some of the best compilers available In all other respects this book should suit all but the most dedicated hackers and is quite reasonably priced

By contrast the 1570/71 disk drives are still too new to have had many books written about them yet I know of only 3, which are in fact editions of the same book in German, American and English The English version is entitled *The Anatomy of the 1571 Disk Drive* and is published by First Publishing If this book had appeared a little later (the German original has been available almost as long as the drives) it would probably have been excellent As it is it is simply very good The main weakness is that in places they have had to anticipate what information users would require and so have not included everything that Commodore forgot All the disk housekeeping commands, file types and special user commands are fully covered, although the inexperienced programmer would probably have appreciated more example programs

This is not, however a book aimed at the novice and over half of it is ROM listings In this case the authors have done a really first class job of adding comments to these and they are genuinely useful Despite this it is still difficult to follow some commands from reception to completion, which is hardly the fault of the authors but is inherent in the byzantine structure of the ROMs There is great potential for getting these C128 disk drives to perform better than intended and this is the type of book you need if you want to try

```
PROGRAM  MFM FORMATTER



10 REM 1570 MFM DISK FORMATTER -
 BY PAUL SCHOFIELD
20 SCNCLR CHAR 1,30,3,"MFM DISK
FORMATTER"
30 CHAR 1,30,4,"=== ==== ======
=="
40 CHAR 1,10,7,"1- KAYPRO II"
50 CHAR 1,10,8,"2  IBM-8 SS"
60 CHAR 1,10,11,"PLEASE ENTER RE
QUIRED FORMAT      "
70 CHAR 1,41,11,"" INPUT F IF F>
2 OR F<1 THEN GO
80 CHAR 1,10,14,"INSERT DISK AND
 ENTER DEVICE NUMBER (8-11)
   "
90 CHAR 1,58,14,"" INPUTD IF D<8
OR D>11 THEN 80
100 IF F=1 THEN B5=2 B7=10 B3=12
9
110 IF F=2 THEN B5=2 B7=8 B3=129
120 CB="UO"+CHR$(D)+CHR$(B3)+CHR
$(0)+CHR$(B5)+CHR$(38)+CHR$(B7)
130 CHAR 1,10,18,"" OPEN 1,D,15,
C$ PRINT#1,"UO"+CHR$(4) GET#1,AS
 CLOSE1
140 SB=ASC(A$)-48 IF DS>1 OR SB<
>0 THEN PRINT "FORMAT ERROR - "
OS$ "     STATUS = ",SB, END
150 PRINT "ANOTHER DISK (Y/N)
",
160 INPUT Y$:IF Y$="Y" OR Y$="Y
Y" THEN CHAR 1,10,16,"
                    " GOTO 130:EL
SE SCNCLR
170 REM FOR 1571 DOUBLE SIDED FO
RMATS CHANGE CHR$(B) TO CHR$(38)
 IN LINE 110
```

Print Master

Having problems designing sprites, characters and screens? Use this program to print grids to help you

If you are designing a game, a business program or a utility you will no doubt at some time be required to design a screen, sprites and even user defined characters. Print Master will help you with your designs by printing out on an MPS 801 or compatible printer a range of four grids. The grids are as follows

1) A Character Designer Grid (row of three)
2) Sprite Designer Grid.
3) Small Screen Grid and
4) Large Screen Matrix

Getting it in

The program is presented as a Basic listing. You should use the *SYNTAX CHECKER* program found on the *LISTINGS* page of this magazine to check each of your lines as you enter the program. Read the *LISTINGS* page for more information on how to do this.

When RUN the program will prompt you for the type of grid that you require and the number of copies of the GRID that you require.

After you have been using the grids for a while they will no doubt become an invaluable aid to your programming. Why not photocopy a number of grids to save wear on your printer ribbon.

```
PROGRAM  PRINTMASTER

S7   50 M$=CHR$(147):N$=CHR$(13
07   55 O$=  :DOWN(5)   P$="":RIGHT(1
     0)
D6   60 AA$=CHR$(151 ZZ$=CHR$(5)
A3  100 PRINTM$   :C7(  :POKE53280
     ,11 POKE53281,0 POKE650,128

49  130 PRINTTAB(13) :CA S*12 CS
    )
C4  135 PRINTTAB(13)  :S= RVSON)
    :SP):INT  :SH):ENU  :RVSOFF S=)
    "
F5  140 PRINTTAB(13) :CZ S*12 CX
    )
2C  145 PRINTCHR$(14):N$ LEFT$(O$
    ,5) LEFT$(P$,2)
16  150 PRINT :C$(11  :SC:HARACTE
    R  :SM:ATRIX
62  155 PRINT :DOWN RIGHT2)2) :S
    $:PRITE  :SM:ATRIX"
42  160 PRINT"(DOWN RIGHT2)3) :S
    $:MALL  :SS:CREEN  :SM:ATRIX"
58  165 PRINT :DOWN RIGHT2)4) :S
    L:ARGE  :SS:CREEN  :SM:ATRIX"
8E  167 PRINT :DOWN RIGHT2)5) :S
    E:XIT
8A  170 PRINTN$,LEFT$(O$ 15) LEF
    T$(P$ 5),
53  175 :INPUT  :SW:HICH  W
65  180 IFW=5THENPRINTM$CHR$(142
    ) END
75  185 IFW<1ORW>4THEN50
F3  190 FORT=1TO500 NEXT
2A  195 ONWGOSUB1000 2000 3000 4
    000
46  200 GOTO50
27  997 REM**********************
    ***********
15  998 REM CHARACTER MATRIX DES
    IGNER      *
65  999 REM**********************
    ***********
46  1000 PRINTM$ LEFT$(O$ 5):LEFT
    $(P$ 3)
18  1010 PRINT :ST:HIS ROUTINE W
    ILL PRINT OUT COPIES
FE  1015 PRINT :RIGHT3:OF A CHAR
    ACTER MATRIX IN ROWS OF 3
48  1030 PRINT :DOWN2 R:IGHT3.SY)
    OU SELECT HOW MANY ROWS TO P
    RINT
BD  1035 PRINT"(RIGHT3)BUT WHATE
    VER YOU CHOOSE WILL BE"
63  1040 PRINT"(RIGHT3)MULTIPLIE
    D BY 3"
66  1050 PRINT :DOWN2 RIGHT3)" :I
    NPUT :SN:UMBER OF ROWS :NR
8C  1055 :FNR<1THEN1000
66  1060 PRINT :DOWN2 RIGHT3 ST:
    HIS WILL GIVE YOU NR*3 MATRI
    XES
40  1070 PRINT"(DOWN2 R:IGHT3)
    :INPUT"(SD:O YOU WISH TO CHAN
    GE THIS  Y:LEFT3)  :YS
57  1080 IFYS="Y ORYS="Y ANDYS<
    > N THEN1000
80  1150 FORC=1TONR
8F  1160 OPEN4,4
68  1168 REM**:SHIFT + O SH:IFT +
    P:*8
C9  1170 A$=  :SO SP,SO SP SO SP
    SO SP SO SP SO SP SO SP SO S
    P:"
AD  1171 ;
49  1174 REM**:SHIFT + L SH:IFT +
    @:*8 :LOGO + @:*4
03  1175 B$=  :SL S@ SL S@ SL S@
    SL S@ SL S@ SL S@ SL S@ SL S
    @: :C@4)
E4  1180 C$="":SPC23)""
FD  1185 D@="2 6 3 1:SPC17)
7E  1190 E@= 0 4 2 6 8 4 2 1   0A
    TA:SPC3)
F0  1200 PRINT#4 AA$
66  1205 PRINT#4 C$C$C$
3E  1210 PRINT#4 D$D$D$
5E  1215 PRINT#4 E$E$E$
34  1220 FORL=1TO8
E2  1225 PRINT#4 AA$A$ :SPC0) A$
    :SPC0) A$ZZ$
FC  1230 PRINT#4 AA$C$ :SPC3) B$
    :SPC3) B$ZZ$
A1  1235 NEXTL
FD  1240 PRINT#4 AA$:CLOSE4:NEXT
    C:RETURN
31  1997 REM**********************
    ***********
49  1998 REM SPRITE MATRIX DESIG
    NER      *
73  1999 REM**********************
    ***********
55  2000 PRINTM$ LEFT$(O$ 5):LEFT
    $(P$ 2)  :ST:HIS ROUTINE WIL
    L PRINT OUT A COPY OF
EF  2005 PRINT :RIGHT2:COPIES OF
     A SPRITE MATRIX DES:IGNER
AB  2010 PRINTN$ LEFT$(O$ 15):LEF
    T$(P$ 2),
3C  2015 INPUT :SH:OW MANY COPIE
    S   NR
C7  2020 IFNR=0THEN2000
B2  2030 FORC=1TONR
```

By G Thorpe

```
10  2050 OPEN4,4
FC  2060 CS=0:I=SPC(8)
EA  2070 DS="2,6,3,1:I=SPC(8)
44  2080 E$="8,4,2,6,8,4,2,1
94  2090 A$="{RVS}REM**{SHIFT + O SHIFT + P}*{OFF}
E5  2100 B$="{SL S$ SL S$ SL S$
```

Program Lock

Are you keen to keep prying eyes off your beautiful Basic coding? This utility will help you to keep a few secrets.

By David Derrick

Here is a program to make it very difficult for prying eyes to look at your Basic programs without your permission. PROG–LOCK is a skeleton program which sits below your own program so that when the program is loaded and listed, the result will be the message 10 SYS2076 which is far from informative.

To see the Basic program you must RUN the program which will then ask you for a password, which if entered correctly will allow you to run or list the Basic program. If you don't give the correct password the machine resets itself!

Getting Going

Type in the program and SAVE it as LOCK LOAD **before** RUNning it. If all the data is correct PROG–LOCK is SAVEd to tape or disk as you require.

Now to keep out the nosey–parkers.

Turn the computer on and off, then LOAD PROG–LOCK back in. Type

POKE43,124

followed by **RETURN**.

Now LOAD in the BASIC program you want to protect, and type

POKE43,1

followed by **RETURN**.

The default password is set as ELEPHANT but you change it by typing

SYS2150

followed by **RETURN** and you can then type in a new password of up to 20 characters (counting RETURN at the end as one character).

That's it! The program is now protected and just needs to be SAVEd.

Note that PROG–LOCK doesn't copy–protect your program and would be rapidly defeated by an experienced hacker, but it will make the casual program thief think twice, not to mention the cachet it will give to your programs!

```
PROGRAM: PROG LOCK

88  10 X=16385:T=0:N=0
39  20 READA:IFA=-1THEN40
88  30 POKEX,A:T=T+A:N=N+1:X=X+1
    :GOTO20
ED  40 IFT<>10932THENPRINT"DATA
    VALUE ERROR":END
0F  50 IFN<>125THENPRINT"DATA AM
    OUNT ERROR":END
52  60 PRINT"{17}ARE OF {D}ISK ?
C7  70 GETD$:IFD$<>"I"ANDD$<>"D"
    THEN70
39  80 POKE251,1:IFD$="D"THENPOK
    E251,8
5C  90 POKE43,1:POKE44,84:POKE45
    ,127:POKE46,64:SAVE"PROG-LOC
    K",PEEK(251)
1D  1000 DATA 11,8,10,0,158,50,4
    8,55,54,0
FB  1010 DATA 0,0,147,8,14,32,20
    0,65,63,83
A6  1020 DATA 67,79,82,68,32,63
    0,169,13,160
12  1030 DATA 8,32,30,171,160,0
    32,207,255,153
3F  1040 DATA 64,3,200,201,13,20
    0,245,169,0,153
A7  1050 DATA 64,3,160,0,185,64
    3,240,12,217
A6  1060 DATA 01,0,240,3,76,226,
    252,200,76,55
90  1070 DATA 0,169,124,133,43,1
    69,6,133,44,96
81  1080 DATA 89,76,69,60,72,65
    78,84,13,0
3D  1090 DATA 153,34,72,89,76,76
    ,79,34,0,106
8E  1100 DATA 0,169,13,160,0,32,
    30,171,160,0
72  1110 DATA 32,207,255,153,61,
    8,200,201,13,208
95  1120 DATA 245,96,0,0,0,-1
```

Want to use the programmes in this publication?

Can't afford the time to type them in?

Why not buy them all on disk or cassette?

Typing in long programs can be a pretty daunting task. Once you've entered the program there will no doubt be typing errors that need to be corrected. Why not save yourself time and trouble by buying a disk or cassette of the programme from this publication. All of the programmes that are presented here are on the cassette or disk at a bargain price of £6.00 for disk or £4.00 for cassette.

The disk and cassette are only available mail order from the address on the order form. A cheque payable to ASP Ltd for the correct amount should be included with the order. Overseas customers should add £1.00 for postage.

ORDER FORM — PLEASE COMPLETE IN BLOCK CAPITALS

NAME	QUANTITY	PRICE	ORDERCODE	TOTAL
SERIOUS USER DISK		£6.00	YSUGD	
SERIOUS USER TAPE		£4.00	YSUGC	
OVERSEAS POSTAGE		£1.00		
			TOTAL	

NAME

ADDRESS

POSTCODE

I enclose a cheque/postal order for £ made payable to ASP LTD for the Your Commodore Serious User Guide Disk/Tape

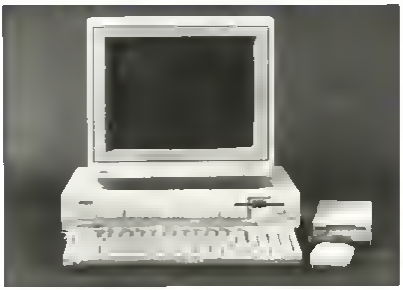
All orders should be sent to Your Commodore, Readers Services, Argus Specialist Publications, 9 Hall Road, Hemel Hempstead, Herts HP2 7BH

Please allow 28 days for delivery

AT LAST!

AN ECONOMICAL ALTERNATIVE TO THE BULKY EXTERNAL AMIGA DISK DRIVES

3.5" EXTERNAL FLOPPY DISK DRIVE FOR THE COMMODORE AMIGA



SPECIAL OFFER £139.00 inc.!

CUMANA CAX 354

- NEC 3.5 inch double
- storage capacity
- Quiet operation
- Lower power consumption
- Connection enables easy addition of 5.25" drives

SPECIFICATION

Track to track 3ms • Settling time 15ms • Rated output Seven 300 Gauss • Data Transfer Rate 125/250 Kbits per sec • Number of tracks 80 • Number of sides 2

FED UP WITH PAYING HIGH PRICES FOR YOUR 5-25" FLOPPY DISKS??? JUST LOOK AT OUR PRICES!!!!

DS/DD 5.25" DISKS AT THE SILLY PRICE OF JUST **£6.00** PER TEN

SAVE EVEN MORE MONEY BUY TWO PACKS AND SAVE ANOTHER **£2.00**

TWO PACKS OF TEN 5.25" DISKS JUST £10.00

DISK NIBBLER

Use the Nibbler only a few times. Save the cost of the Nibbler with just one hundred disks you will have purchased. Only **£5.00** or **FREE** if you buy 50 or more 5.25" disks.

SPECIAL OFFER

50 5.25" disks	£25.00
100 5.25" disk box	£10.50
Disk Nibbler	F.O.C
OUR NORMAL PRICE	£48.50
OFFER PRICE	£35.00
SAVE	£11.50

Prices include VAT and delivery UK mainland

COMMODORE CABLES

CENTRONICS PRINTER CABLE
For the C64/128 user, good for cartridges or other cable. The cable is supplied with a free lead switch for the Epson range of printers. Works with all word processing packages. Centronics printers. **ONLY £10.00** incl

SERIAL EXTENSION CABLES
Extension cable for commodore or other disk drive cable by up to 2 metres
Male to female extension cable **£5.00** inc
Male to male extension cable **£7.00** incl

C64/128D KEYBOARD EXTENSION
Do you find you self-destruct clearly by the short keyboard cable on the C128D. Solve your problem with our 1 metre extension cable
We have mounted our C128 under the desk to save room
SPECIAL OFFER PRICE ONLY £15.00 incl

AT LAST!!!! 3.5" DISKS AT SENSIBLE PRICES

Double sided double density 3.5" verbatim disks

ONLY £16.00 for pack of ten disks
SAVE EVEN MORE MONEY!!!!
BUY TWO PACKS FOR ONLY £30.00

Please add one extra disk for every two disks ordered to cover postage

NASHA DS/DD 3.5" DISKS BOXED WITH LABEL
OUR LOW PRICE **£23.00** per box ten
SAVE EVEN MORE MONEY!!!
BUY TWO BOXES FOR ONLY **£44.00**

We believe our prices are the lowest around
All prices include VAT and UK postage

LOCKABLE DISK BOXES

DB3/00
3.5" disk box holds 90 disks. Lockable, first class value at only **£12.50** or only **£11.50** when you buy 10 or more 3.5" disks

DB5/70
5.25" disk box holds 70 disks great value only **£9.50** or only **£8.50** when you buy 20 or more 5.25" disks

DB5/100
5.25" disk box holds 100 disks. Large in at only **£11.50** or only **£10.50** when you buy 20 or more 5.25" disks

---

NEW! NEW! NEW! NEW! NEW!

COMMODORE C64/C128 RS232 INTERFACE

AT LAST! RS232 interface that will not cost you the earth.
The H&P Computers Commodore RS232 Interface is a fully intelligent standard RS232 interface with all handshaking lines that plugs into the user port.
Will fit all modems and printers with a 25 way D connector.
Up till now you would have had to pay between **£35.00** and **£50.00** for a RS232 interface for the C64/C128.
The H&P Computers RS232 is only **£25.00** incl. and we even give you an extra modem comms program on disk **free of charge**.

ONCE AGAIN WE BRING THE BEST FOR LESS
ONLY £25.00 INCL

H&P COMPUTERS UK,
9 HORNBEAM WALK, WITHAM, ESSEX CM8 2SZ. Tel: (0376) 511471

TOTAL BACK-UP POWER C64/128

PERIPHERALS..THE FINAL FRONTIER..OUR MISSION..TO BOLDLY GO WHERE NO OTHER UTILITIES HAVE GONE BEFORE

CAPTAINS LOG...THE TOTAL SOLUTION TO ALL YOUR BACK-UP NEEDS...THE ULTIMATE BACK-UP CARTRIDGE HERE NOW !!

REPORT ON FINDINGS

Action Replay Mk III is more powerful, more friendly and will back up more programs than any competing utility by taking a "Snapshot" of the program in memory so it doesn't matter how it was loaded... from disk or tape, all normal or turbo speeds... the results are the same – **Perfect!!! Amazing!!!**

STARBASE UPDATE

- Simple to use- just press the button and make a complete backup. Tape to Tape, Tape to Disk, Disk to Disk, Disk to Tape. – THE PROCESS IS AUTOMATIC – JUST GIVE THE BACKUP A NAME.
- All backups will reload at turbo speed independently of the cartridge.
- Dual speed tape turbo system. Programs can load up to 3 times more than normal turbo – that's over 10 times normal Commodore speed
- Freeze the Action then view the program with the monitor feature. Add pokes for infinite lives etc. Then Action the game or backup – ideal for customised versions of your games.
- Picture Save. Save any multi-colour Hires screen to disk or tape. Compatible with Blazing Paddles, Koala, Slideshow etc.
- Fully compatible with 1541, 1541C, 1570, 1571, and whatever or any CBM compatible disk recorder
- For C64, 64C, 128, 128D (in 64 mode).
- Unique Sprite Monitor. Freeze the Action and view all the Sprites, watch the animations scroll across the screen. Save Sprites to disk or tape. Customise your games by loading sprites from one press to another – then Action the program or make a backup.
- Compatible with and DOS two Turbo ROM systems.
- Backup process in turbo speed – faster than any rivals.
- Special compacting techniques. Each program is saved as a single file.
- Transfers multistage tape programs to disk – more than any other cartridge – even the extra stages are turbo load – a unique feature.
- Sprite Killer! make yourself indestructible by disabling Sprite collisions in games.
- Fast load, format (10 secs).
- Built-in unstoppable reset button.

ONLY £29.99 POST FREE

ACTION REPLAY MK III

PLUS Built In

FASTLOADER

Action Replay III even has a built in disk fast loader which speeds up loading 5 times. Uses no memory – invisible to the system. You could pay £20 alone for this feature

WARP 25

BREAKS THROUGH THE 10 SECOND BARRIER!

Action Replay II now comes with an amazing, ultra-fast Basic/Kernal that will reload your backups at TWENTY FIVE TIMES normal speed. The fastest disk turbo yet devised!! There are NO CATCHES WARP 25 works with ALL your games. Works with any disk drive. No-printer required. No hardware modifications necessary. No user knowledge required – programs load AUTOMATICALLY. LOADING TIME 0.8 SECONDS! For 6 tightest games saved in WARP 25 in conjunction with ACTION REPLAY III. This time is the COMPLETE load screens from start to finish. Reload is entirely INDEPENDENT of the cartridge or any other hardware. Compare these (accurate!) figures for some rival backup systems:

SYSTEM	LOAD TIME		
ACTION REPLAY MK III	25 SECS	THREE	NO
ACTION REPLAY MK II	9.0 SECS	THREE	NO
FREEZE FRAME (MK IV)	30 SECS	TWO OR THREE	NO
FREEZE FRAME (LAZER)	[illegible] SECS	TWO	YES
EXPERT SYSTEM	[illegible] SECS	THREE	NO

All purchasers of Action Replay III will receive WARP 25 FREE with their cartridge. Existing Action Replay owners can obtain WARP 25 Disk turbo by sending £25.00 post free (No need to send your cartridge). OR obtain it FREE on the Enhancement Disk (£7.00).

THE ACTION REPLAY ENHANCEMENT DISK

[illegible] Wrestling, Superbase, Master Notebook, Wordcruncher, [illegible] parameter system. No [illegible] throughout. NOTE: Standard [illegible] Boulderdash, Impossible Mission, Monty Comes at, etc. Blue [illegible] FREE Multicolour Sidebars for display of loading [illegible] etc. saved by Action Replay. Great

PERFORMANCE GUARANTEE

[illegible] Action Replay Mk III [illegible] any other cartridge [illegible] Commodore [illegible] Action Replay [illegible] Mk III [illegible] Mk III [illegible] Mk III